**File Handling**

**File(IO Package)**

Agenda:

**1. File**

**2. FileWriter**

**3. FileReader**

**4. BufferedWriter**

**5. BufferedReader**

File:

File f=new File("abc.txt");

This line 1st checks whether abc.txt file is already available (or) not, if it is already available then "f" simply refers that file. If it is not already available then it won't create any physical file just creates a java File object represents name of the file.

**Example:**

import java.io.\*;

class FileDemo{

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

File f=new File("abc.txt");

System.out.println(f.exists());//false

f.createNewFile();

System.out.println(f.exists());//true

}

}

1st run

* During the first run there is not any file so the program is giving **false**
* After running the method **CreateNewFile**() with the object f of File , it will create the file in the location and when we check again if the file exists or not it will give **true**.

2nd run

* Now if again run the program it will first check if the file is present or not. As the Program is executed previously so the file already present so it will give **true**
* In the last Line it is giving true as the file is present.

=> A java File object can represent a directory also.

**Example:**

import java.io.File;

import java.io.IOException;

class FileDemo{

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

File f=new File("cricket123");

System.out.println(f.exists());//false

f.mkdir();//Creates a new directory

System.out.println(f.exists());//true

}

}

1st run

* During the first run there is not any directory present so the program is giving **false**
* After running the method **mkdir ()** with the object f of File , it will create the directory in the location and when we check again if the file exists or not it will give **true**.

2nd run

* Now if again run the program it will first check if the directory is present or not. As the Program is executed previously so the directory already present so it will give **true**
* In the last Line it is giving true as the file is present.

**Note**: In UNIX everything is a file, java "file IO” is based on UNIX operating system hence in java also we can represent both files and directories by File object only.

**Constructors of File class**

**File f=new File(String fname)**

**File f=new File(String directoryName,String fileName);**

**File f=new File(File f,String fileName);**

**File class constructors**

1. File f=new File(String name);

=> Creates a java File object that represents name of the file or directory in current working directory.

**Example:-**

**1. File f=new File("abc.txt");**

**2. File f=new File(String subdirname,String name);**

=> Creates a File object that represents name of the file or directory present

in specified sub directory.

eg#1. File f1=new File("abc");

f1.mkdir();

File f2=new File("abc","demo.txt");

f2.createNewFile();

In the above example the directory should be present in order to create a file in it, otherwise it will throw an IO Exception.

3. File f=new File(File subdir,String name);

eg#1.File f1=new File("abc");

f1.mkdir();

File f2=new File(f1,"demo.txt");

f2.createNewFile();

Requirement

**Q:- Write code to create a file named with demo.txt in current working directory.**

cwd

|=> abc.txt

Program:

import java.io.\*;

class FileDemo{

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

File f=new File("demo.txt");

f.createNewFile();

}

}

Requirement

**Q:-Write code to create a directory named with IPLTeam in current working directory and create a file named with abc.txt in that directory.**

cwd

|=> IPLTeam

|=> abc.txt

Program:

import java.io.\*;

class FileDemo{

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

File f1=new File("IPLTeam");

f1.mkdir();

File f2=new File("IPLTeam","abc.txt");

f2.createNewFile();

}

}

Requirement

**Q: Write code to create a file named with rcb.txt present in D:\IPLTeam**

**folder.**

D

|=> IplTeam

|-> rcb.txt

Program:

import java.io.\*;

class FileDemo{

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

File f=new File("C:\\Users\\gauta\\eclipse-workspace\\FileIo\\ IPLTeam","rcb.txt");

f.createNewFile();

}

}

Assuming C:\\IPLTeam should be already available otherwise it would result in

"FileNotFoundException".

Important methods of File class

**1. boolean exists();**

**2. boolean createNewFile()**

**3. boolean mkdir()**

**4. boolean isFile();**

**5. boolean isDirectory()**

**6. String[] list();**

**7. long length();**

**8. boolean delete()**

Important methods of file class:

**1. boolean exists();**

Returns true if the physical file or directory available.

**2. boolean createNewFile();**

This method 1st checks whether the physical file is already available or not if it is already available then this method simply returns false without creating any physical file. If this file is not already available then it will create a new file and returns true

**3. boolean mkdir();**

This method 1st checks whether the directory is already available or not if it is already available then this method simply returns false without creating any directory. If this directory is not already available then it will create a new directory and returns true.

**4. boolean isFile();**

Returns true if the File object represents a physical file.

**5. boolean isDirectory();**

Returns true if the File object represents a directory.

**6. String[] list();**

It returns the names of all files and subdirectories present in the specified directory.

**7. long length();**

Returns the no of characters present in the file.

**8. boolean delete();**

To delete a file or directory

**Requirement**

**Q: Write a program to display the names of all files and directories**

**present in D:\EnterpriseJava**

**Requirement**

**Q: Write a program to display only file names.**

**Requirement**

**Q: Write a program to display only directory names.**

**Requirement**

**Q: Write a program to display the names of all files and directories**

**Present in C:\\Users\\gauta\\eclipse-workspace\\FileIo**

import java.io.File;

import java.io.IOException;

public class TestApp {

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

File f=new File("C:\\Users\\gauta\\eclipse-workspace\\FileIo ");

String []s = f.list();

int count =0;

for(String si:s){

System.out.println(si)

count++

}

System.out.println(“no.of file is”,count);

}

**Requirement**

**Q: Write a program to display only file names.**

import java.io.File;

import java.io.IOException;

public class TestApp {

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

File f=new File("C:\\Users\\gauta\\eclipse-workspace\\FileIo ");

String[] s= f.list();

int count=0;

for(String s1:s){

File f1=new File(f,s1);

if (f1.isFile()){

count++;

System.out.println(s1);

}

}

System.out.println("The no of Directories are :: "+count);

}

}

**Requirement Q: Write a program to display only directory names**

import java.io.File;

import java.io.IOException;

public class TestApp {

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

File f=new File("D:\EnterpriseJava");

String[] s= f.list();

int count=0;

for(String s1:s){

File f1=new File(f,s1);

if (f1.isDirectory()){

count++;

System.out.println(s1);

}

}

System.out.println("The no of Directories are :: "+count);

}

}

**FileWriter**

FileWriter is generally used to perform write operation in the file. Till now we have only read how to create a file and directory but haven’t perform any operation related to the text file i.e. writing text in the text file. So in this section we will perform the writing operation in the file let’s see how to do the writing operation.

🡺writeOperation

**Constructors:**

**FileWriter fw=new FileWriter(String name);**

**FileWriter fw=new FileWriter(File f);**

The above 2 constructors meant for overriding the data to the file. Instead of overriding if we want append operation then we should go for the following 2 constructors.

**FileWriter fw=new FileWriter(String name,boolean append);**

**FileWriter fw=new FileWriter(File f,boolean append);**

If the specified physical file is not already available then these constructors will create that file.

**Methods:**

**1. write(int ch);**

To write a single character to the file.

**2. write(char[] ch);**

To write an array of characters to the file.

**3. write(String s);**

To write a String to the file.

**4. flush();**

To give the guarantee the total data include last character also written to the file.

**5. close();**

To close the stream.

**Example 0:-**

import java.io.FileWriter;

import java.io.IOException;

public class FileWriter{

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

FileWriter fs = new FileWriter(“info.txt”);

fs.write(“72”);

fs.write(“neuron\nTechnology\n”); \\ “\n will change the line in the file.

Char[] ch={“a”,”b”,”c”,”d”};

fs.write(ch);

fs.flush();

fs.close();

}

}

In **FileWriter**, It automatically creates a file unlike file where we do create a file using different method, here we can simple create a file using the object of the **FileWriter**

**eg#1**.

import java.io.FileWriter;

import java.io.IOException;

public class TestApp {

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

FileWriter fw=new FileWriter("abc.txt");

fw.write(73);

fw.write("neuron\nTechnology\nPrivate\nLimited");

fw.write("\n");

char ch[] ={'a','b','c'};

fw.write(ch);

fw.flush();

fw.close();

}

}

A new file will be created automatically abc.txt

Ineuron

Technology

Private

Limited

Abc

**Note:**

=> The main problem with FileWriter is we have to insert line separator

manually,which is difficult to the programmer. ('\n')

=> And even line separator varing from system to system.

=>Represenation of "\n" would vary from system to system.

**FileReader:**

=> By using FileReader object we can read character data from the file.

**Constructors:**

**FileReader fr=new FileReader(String name);**

**FileReader fr=new FileReader (File f);**

**Methods**

**1. int read();**

It attempts to read next character from the file and return its Unicode value. If the next character is not available then we will get -1.

**2. int i=fr.read();**

**3. System.out.println((char)i);**

As this method returns unicodevalue , while printing we have to perform type casting.

**4. int read(char[] ch);**

It attempts to read enough characters from the file into char[] array and returns the no of characters copied from the file into char[] array.

**5. File f=new File("abc.txt");**

**6. Char[] ch=new Char[(int)f.length()];**

**7. void close();**

eg#1.

import java.io.FileReader;

import java.io.IOException;

public class TestApp {

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

FileReader fr=new FileReader("abc.txt");

int i=fr.read();

while(i!=-1){ // when there is no character the value will be -1

System.out.println((char)i);

i=fr.read();

}

}

From the above code we can say that FileReader can read character one by one so it is time consuming activity so let’s suppose if we use array to store the value of file and then read it . It will be more beneficial let’s see how:-

**FileReader fr=new FileReader("abc.txt");**

**Char [] ch = new Char[1000];**

**fr.read(ch);**

**for(char c:ch)**

**sysout(c);**

**fr.close();**

so this is the way we can read the file at once by using array.

**eg#2. Reading an array of characters**

abc.txt

1000 characters are available

Scenario1:

FileReader fr=new FileReader("abc.txt");

char[] ch=new char[10];

int noOfCharactersCopied=fr.read(ch);

Scenario2:

FileReader fr=new FileReader("abc.txt");

char[] ch=new char[10000];

int noOfCharactersCopied=fr.read(ch);

import java.io.FileReader;

import java.io.IOException;

import java.io.File;

public class TestApp {

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

File f=new File("abc.txt");

FileReader fr=new FileReader(f);

char ch[] = new char[(int)f.length()];

fr.read(ch);

String data=new String(ch);

System.out.println(data);

fr.close();

}

}

Usage of FileWriter and FileReader is not recommended because of following reasons:-

1. While writing data by FileWriter compulsory we should insert line separator(\n)

manually which is a bigger headache to the programmer.

2. While reading data by FileReader we have to read character by character instead

of line by line which is not convenient to the programmer.

Assume we need to search for a 10 digit mobile no present in a file called "mobile.txt"

=>Since we can read only character just to search one mobile no 10 searching and

to search 10,000 mobile no we need to read 1cr times, so performance is very low.

**3. To overcome these limitations we should go for BufferedWriter and BufferedReader**

**concepts.**

**BufferedWriter:**

It can't communicate with the file directly, it can communicate only with writer Object.

**Constructor**

**BufferedWriter bw=new BufferedWriter(Writer w);**

**BufferedWriter bw=new BufferedWriter(Writer w,int buffersize);**

**Which of the following declarations are valid?**

1. BufferedWriter bw=new BufferedWriter("cricket.txt"); //invalid

2. BufferedWriter bw=new BufferedWriter (new File("cricket.txt")); //invalid

3. BufferedWriter bw=new BufferedWriter (new FileWriter("cricket.txt")); //valid

4. BufferedWriter bw=new BufferedWriter(new BufferedWriter(new

FileWriter("crickter.txt"))); //valid

**Methods**

**1. write(int ch);**

**2. write(char[] ch);**

**3. write(String s);**

**4. flush();**

**5. close();**

**6. newLine();**

Inserting a new line character to the file.

**When compared with FileWriter which of the following capability(facility) is available as method in BufferedWriter.**

**1. Writing data to the file.**

**2. Closing the writer.**

**3. Flush the writer.**

**4. Inserting newline character.**

Answer: 4(newLine())

import java.io.\*;

class TestApp

{

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

{

BufferedWriter bw = new BufferedWriter(new FileWriter("abc.txt"));

bw.write(73);

bw.write("neuron");

bw.newLine();

bw.write("technology");

bw.newLine();

char ch[] = {'a','b','c'};

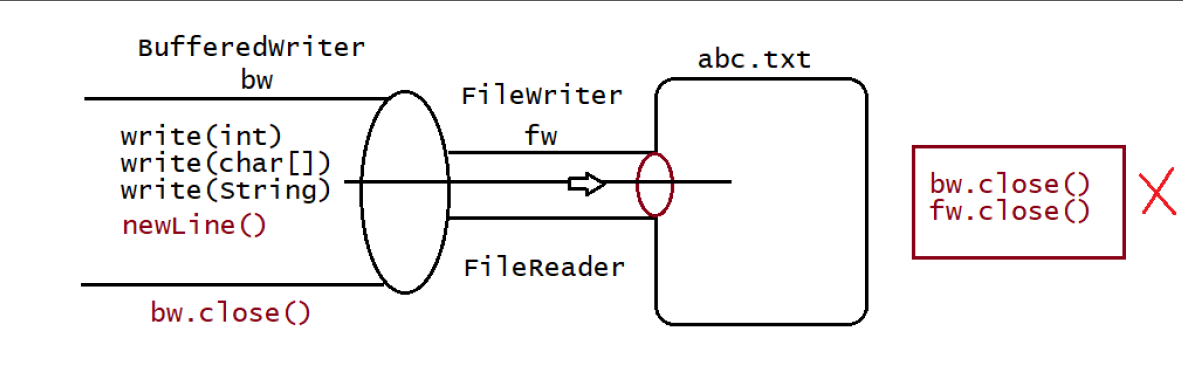
bw.write(ch);

bw.flush();

bw.close();

}

}



**Note**

1.bw.close()// recomended to use

2.fw.close()// not recomended to use

3.bw.close()// not recomended to use

fw.close()

=> When ever we are closing BufferedWriter automatically underlying writer will be

closed and we are not close explicitly.

**BufferedReader:**

This is the most enhanced(better) Reader to read character data from the file.

Constructors:

BufferedReader br=new BufferedReader(Reader r);

BufferedReader br=new BufferedReader(Reader r,int buffersize);

**Note**

=> BufferedReader can not communicate directly with the File it should communicate via some Reader object.

=> The main advantage of BufferedReader over FileReader is we can read data line by line instead of character by character.

**Methods:**

**1. int read();**

**2. int read(char[] ch);**

**3. String readLine();**

It attempts to read next line and return it , from the File. if the next line

is not available then this method returns null.

4. void close();

eg#1.Read the data from the file called "abc.txt"

import java.io.FileReader;

import java.io.IOException;

import java.io.BufferedReader;

public class TestApp {

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

FileReader fr=new FileReader("abc.txt");

BufferedReader br=new BufferedReader(fr);

String line= br.readLine();

while(line!=null){

System.out.println(data);

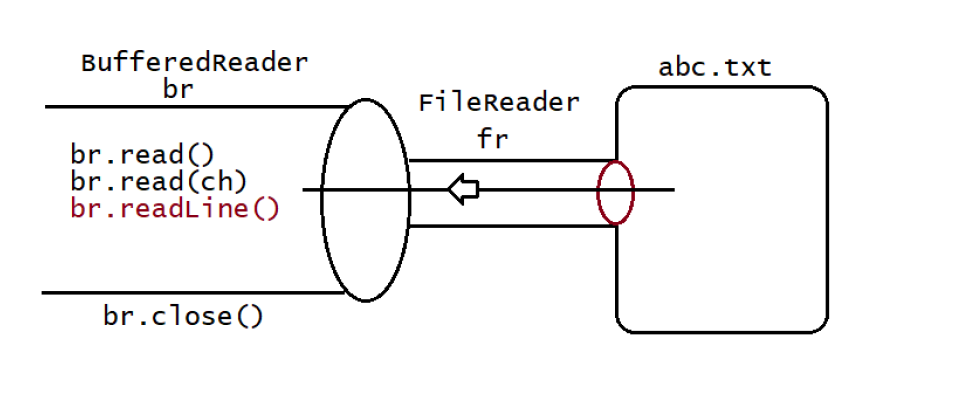
line=br.readLine();

}

br.close();

}

}



**Note:**

**1.br.close()// recomended to use**

**2.fw.close()// not recomended to use**

**3.br.close()// not recomended to use**

**fw.close()**

=> Whenever we are closing BufferedReader automatically underlying FileReader will be closed it is not required to close explicitly.

=> Even this rule is applicable for BufferedWriter also.

**PrintWriter**:

=> This is the most enhanced Writer to write text data to the file.

=> By using FileWriter and BufferedWriter we can write only character data to the File but by using PrintWriter

we can write any type of data to the File.

Constructors:

**PrintWriter pw=new PrintWriter(String name);**

**PrintWriter pw=new PrintWriter(File f);**

**PrintWriter pw=new PrintWriter(Writer w);**

Methods:

**1. write(int ch);**

**2. write (char[] ch);**

**3. write(String s);**

**4. flush();**

**5. close();**

**6. print(char ch);**

**7. print (int i);**

**8. print (double d);**

**9. print (boolean b);**

**10.print (String s);**

**11.println(char ch);**

**12.println (int i);**

**13.println(double d);**

**14.println(boolean b);**

**15.println(String s);**

**Note 1:**

1. The most enhanced Reader to read character data from the File is BufferedReader.

2. The most enhanced Writer to write character data to the File is PrintWriter.

**Note 2:**

1. In general we can use Readers and Writers to handle character data. Where as we can use InputStreams and OutputStreams to handle binary data(like images, audio files, video files etc).

2. We can use OutputStream to write binary data to the File and we can use InputStream to read binary data from the File

Character Data => Reader and Writer

Binary Data => InputStream and OutputStream

**Note 3: Abstract class(AC)**

**Object**

**|**

**| |**

**Writer(AC) Reader(AC)**

| |

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

| | | | |

Outputstream BufferedWriter PrintWriter InputStreamReader BufferedReader

Writer

| |

FileWriter FileReader

**Requirement => file1.txt ,file2.txt copy all the contents to file3.txt**

import java.io.\*;

class TestApp {

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

PrintWriter pw =new PrintWriter("file3.txt");

//copy from file1.txt to file3.txt

BufferedReader br=new BufferedReader(new FileReader("file1.txt"));

String line = br.readLine();

while(line!=null){

pw.println(line);

line = br.readLine();

}

//copy from file2.txt to file3.txt

br=new BufferedReader(new FileReader("file2.txt"));

line = br.readLine();

while(line!=null){

pw.println(line);

line = br.readLine();

}

//closing the resources

pw.flush();

br.close();

pw.close();

}

}

**Requirement => file1.txt file2.txt copy one line from file1.txt and from file2.txt to file3.txt.**

import java.io.\*;

class TestApp

{

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

{

PrintWriter pw =new PrintWriter("file3.txt");

//copy from file1.txt to file3.txt

BufferedReader br1=new BufferedReader(new FileReader("file1.txt"));

BufferedReader br2=new BufferedReader(new FileReader("file2.txt"));

String line1= br1.readLine();

String line2= br2.readLine();

while(line1!=null || line2!=null)

{

if (line1!=null){

pw.println(line1);

line1= br1.readLine();

}

if(line2!=null){

pw.println(line2);

line2= br2.readLine();

}

}

//closing the resources

pw.flush();

br1.close();

br2.close();

pw.close();

}

}

Requirement => Write a program to perform extraction of mobile no only if there is

no duplicates

import java.io.\*;

class TestApp

{

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

{

PrintWriter pw =new PrintWriter("output.txt");

//copy from file1.txt to file3.txt

BufferedReader br1 =new BufferedReader(new FileReader("input.txt"));

String line = br1.readLine();

BufferedReader br2 =null;

while(line!=null)

{

boolean isAvailable = false;

br2=new BufferedReader(new FileReader("delete.txt"));

String target = br2.readLine();

while(target!=null)

{

if(line.equals(target))

{

isAvailable = true;

break;

}

target = br2.readLine();

}

if (isAvailable==false)

{

pw.println(line);

pw.flush();//flush to ensure all data is written to the

file

}

line = br1.readLine();

}

//closing the resources

br1.close();

br2.close();

pw.close();

}

}

Requirement => Write a program to remove duplicates from the file

import java.io.\*;

class TestApp

{

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

{

PrintWriter pw =new PrintWriter("output.txt");

//copy from file1.txt to file3.txt

BufferedReader br1 =new BufferedReader(new FileReader("input.txt"));

String line = br1.readLine();

BufferedReader br2 =null;

while(line!=null)

{

boolean isAvailable =false;

br2 =new BufferedReader(new FileReader("output.txt"));

String target = br2.readLine();

while(target!=null)

{

if (line.equals(target))

{

isAvailable = true;

break;

}

target = br2.readLine();

}

if(isAvailable==false){

pw.println(line);

pw.flush();

}

line = br1.readLine();

}

//closing the resources

br1.close();

br2.close();

pw.close();

}

}

**Serialization**

**Agenda**

1. Serialization

2. Deserialization

3. Transient keyword

4. Static Vs transient

5. Transient Vs final

6. Object graph in serialization.

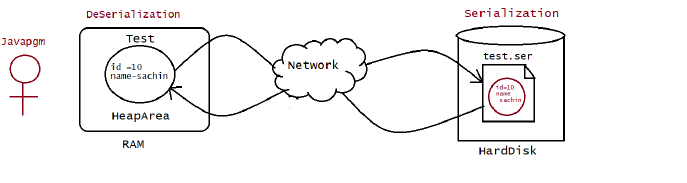
7. customized serialization.

8. Serialization with respect inheritance.

9. Externalization

10. Difference between Serialization & Externalization

11. SerialVersionUID



🡺Suppose we have an object test in the heap area. So what is the life of the object data? As long as program is executing, the object would be available on RAM, once the program stops its execution then the object would be flushed.

🡺Now suppose we want that object to be transfer after some time then what need to be done we need to store in some file so that it can be used after some time let’s see how we are going to store that object in the file.

🡺Now here is the problem, the objects contain data in the form of different datatype so we need to convert that object in such a way that it get stored in the file.

🡺The process of converting the object into file supported / network supported form is called Serialization and reverse of it is called Deserialization.

**Serialization: (1.1 v)**

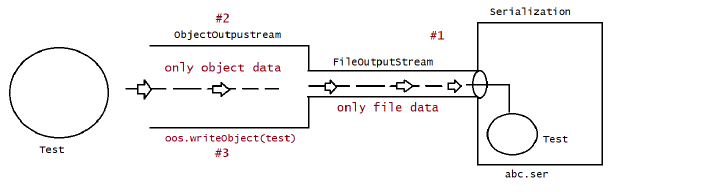
=> The process of saving (or) writing state of an object to a file is called serialization but strictly speaking it is the process of converting an object from java supported form to either network supported form (or) file

supported form.

=> By using **FileOutputStream** and **ObjectOutputStream** classes we can achieve serialization process.

|=> writeObject(Object obj)

Ex: using flipkart booking an iPhone and iPhone reaching to the user.



**Rules:-**

🡺First of all we need to create FileOutputStream, which will create the file abc.ser. After creating FileOutputStream that creates the file.

**Dog d = new Dog();**

**FileOutputStream fos = new FileOutputStream(“abc.ser”);**

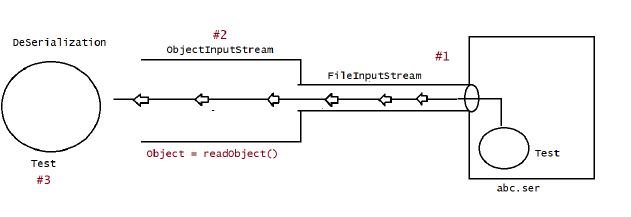
🡺We need to attach that FileOutputStream reference to ObjectOutputStream Object. Now the third steps is to

**ObjectOutputStream oos = new ObjectOutputStream(fos);**

🡺Write the object of class that you want to serialized.

**oos.writeObject(d)**

**De-Serialization:**

****

=> The process of reading state of an object from a file is called DeSerialization but strictly speaking it is the process of converting an object from file supported form (or) network supported form to java supported form.

=> By using **FileInputStream** and **ObjectInputStream** classes we can achieve DeSerialization.

|=> readObject()

**Rules:-**

🡺First of all we need to create FileInputStream, to fetch the file abc.ser. After creating FileInputStream which fetch the file,

**FileInputStream ios = new FileInputStream(“abc.ser”);**

🡺We need to attach that FileInputStream reference to ObjectInputStream Object. Now the third steps is to

**ObjectInputStream Inos = new ObjectInputStream(ios);**

🡺now we type cast the fetch the value in object form

**Dog d =(Dog)ioos.writeObject();**

**Ex Program:-**

eg#1.

import java.io.\*;

class Dog implements Serializable{

int i=10;

int j=20;

}

public class TestApp {

public static void main(String[] args)throws

IOException,ClassNotFoundException {

Dog d1=new Dog();

System.out.println("serialization started");

FileOutputStream fos= new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(d1);

System.out.println("Serialization ended");

System.out.println("Deserialization started");

FileInputStream fis=new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

Dog d2=(Dog) ois.readObject();

System.out.println("Deserialization ended");

System.out.println("Dog object data");

System.out.println(d2.i+"\t" +d2.j);

}

}

**Example2:-**

import java.io.\*

class Dog implements Serializable{

int i=10;

int j=20;

}

class Cat implements Serializable{

int i=100;

int j=200;

}

public class TestApp {

public static void main(String[] args)throws

IOException,ClassNotFoundException {

Dog d1=new Dog();

Cat c1=new Cat();

System.out.println("serialization started");

FileOutputStream fos= new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(d1);

oos.writeObject(c1);

System.out.println("Serialization ended");

System.out.println("Deserialization started");

FileInputStream fis=new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

Dog d2=(Dog) ois.readObject();

Cat c2=(Cat) ois.readObject();

System.out.println("Deserialization ended");

System.out.println("Dog object data");

System.out.println(d2.i+"\t" +d2.j);

System.out.println("Cat object data");

System.out.println(c2.i+"\t" +c2.j);

}

}

**Note:**

1. We can perform Serialization only for Serializable objects.

2. An object is said to be Serializable if and only if the corresponding class implements Serializable interface.

3. Serializable interface present in java.io package and does not contain any methods. It is marker interface. The required ability will be provided automatically by JVM.

4. We can add any no. Of objects to the file and we can read all those objects from the file but in which order we wrote objects in the same order only the objects will come back. That is order is important.if

there is a mismatch in order it would result in "**ClassCastException**".

5. If we are trying to serialize a non-serializable object then we will get **RuntimeException** saying "**NotSerializableException**"

**Transient keyword:**

1. Transient is the modifier applicable only for variable, but not for classes and methods.

2. While performing serialization if we do not want to save the value of a particular variable to meet security constant such type of variable, then we should declare that variable with "transient" keyword.

3. At the time of serialization JVM ignores the original value of transient variable and save default value to the file.

4. That is transient means "not to serialize".

**Example:-**

Class Human implements Serializable

{

int Account =343534634;

transient int psw =243535345;

}

Class test {

Public static void main(String[] args) throws FileNotFound,IOExceptions{

Human h =new Human();

Sysout(“Serialization started”);

FileOutputStream fos = new FileOutputStream(“human.ser”);

ObjectOutputStream oos =new ObjectOutputStream(fos);

oos.writeObject(h);

Sysout(“Serialization Ended”);

FileInputStream Ios = new FileInputStream(“human.ser”);

ObjectInputStream ioos =new ObjectInputStream(Ios);

Human h =(Human)ioos.readObject();

Sysout(h.Account+”=====>”h.psw);

}

}

Output:-

Serialization started

Serialization Ended

343534634=====>0

As we can see here, variable is not present so some time there is need to hide data of variable while serializing at that time transient access modifier is used with variable.

This access modifier is only used with variable.

**Static vs Transient**

Now suppose if make **Transient** variable static then what will happen?

As we know whenever we static keyword with variable it becomes the part of the class not the part of the object. So whenever we make any transient variable static, it didn’t participate in the serialization as it is the part of class and it can be accessed directly through the class name.

Class Human implements Serializable

{

int Account =343534634;

static transient int psw =243535345; //static transient variable

}

Class test {

Public static void main(String[] args) throws FileNotFound,IOExceptions{

Human h =new Human();

Sysout(“Serialization started”);

FileOutputStream fos = new FileOutputStream(“human.ser”);

ObjectOutputStream oos =new ObjectOutputStream(fos);

oos.writeObject(h);

Sysout(“Serialization Ended”);

FileInputStream Ios = new FileInputStream(“human.ser”);

ObjectInputStream ioos =new ObjectInputStream(Ios);

Human h =(Human)ioos.readObject();

Sysout(h.Account+”=====>”h.psw);

}

}

Output:-

Serialization started

Serialization Ended

343534634=====>243535345

**Transient vs Final**

Whenever we use final with transient access modifier we make that variable constant, that mean it directly participated with the value, it didn’t participate in serialization.

eg:

final int x= 10;

int y = 20;

System.out.println(x);// compiler will replace this as System.out.println(20)

becoz x is final.

System.out.println(y);

Class Human implements Serializable

{

int Account =343534634;

final transient int psw =243535345; //final transient variable

}

Class test {

Public static void main(String[] args) throws FileNotFound,IOExceptions{

Human h =new Human();

Sysout(“Serialization started”);

FileOutputStream fos = new FileOutputStream(“human.ser”);

ObjectOutputStream oos =new ObjectOutputStream(fos);

oos.writeObject(h);

Sysout(“Serialization Ended”);

FileInputStream Ios = new FileInputStream(“human.ser”);

ObjectInputStream ioos =new ObjectInputStream(Ios);

Human h =(Human)ioos.readObject();

Sysout(h.Account+”=====>”h.psw);

}

}

Output:-

Serialization started

Serialization Ended

343534634=====>243535345

Declaration output

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

1.

int i=10;

int j=20;

output:: 10 20

2.

transient int i=10;

int j=20;

output:: 0 20

3.

transient int i=10;

transient static int j=20;

output:: 0 20

4.

transient final int i=10;

transient int j=20;

output:: 10 0

5.

transient final int i=10;

transient static int j=20;

output: 10 20

**Note:**

We can serialize any no of objects to the file but in which order we serialized in the same order only we have to deserialize, if we change the order then it would result in "ClassCastException".

Example :

class Dog1 implements Serializable{

int a=6;

int b=8;

int c=9;

}

class Monkey implements Serializable{

int a=10;

int b=11;

int c=12;

}

class Rat1 implements Serializable{

int a=23;

int b=23;

int c=3;

}

public class serial4 {

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

Dog1 d = new Dog1();

Monkey m = new Monkey();

Rat1 r = new Rat1();

FileOutputStream fos = new FileOutputStream("gsgds.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d);

oos.writeObject(m);

oos.writeObject(r);

FileInputStream inos = new FileInputStream("C:\\Users\\gauta\\eclipse-workspace\\Serializations\\gsgds.ser");

ObjectInputStream ioos = new ObjectInputStream(inos);

Rat1 ha =(Rat1)ioos.readObject();

Dog1 da =(Dog1)ioos.readObject();

Monkey ma=(Monkey)ioos.readObject();

System.***out***.println(ha.a +"====>"+ha.b+"==>"+ha.c);

System.***out***.println(da.a +"====>"+da.b+"==>"+da.c);

System.***out***.println(ma.a +"====>"+ ma.b+"==>"+ ma.c);

Output :-

It is throwing an exception as the sequence of serialisation and deserialization is not same. To solve this problem we need to use **Reflection class.** Reflection class gives the information about the class at the runtime.

Let’s see how to implement Reflection class:-

We need to use this object class for reading the class at runtime:-

**Object obj=ioos.readObject();**

import java.io.\*;

class Dog1 implements Serializable{

int a=6;

int b=8;

int c=9;

}

class Monkey implements Serializable{

int a=10;

int b=11;

int c=12;

}

class Rat1 implements Serializable{

int a=23;

int b=23;

int c=3;

}

public class serial4 {

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

Dog1 d = new Dog1();

Monkey m = new Monkey();

Rat1 r = new Rat1();

FileOutputStream fos = new FileOutputStream("gsgds.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d);

oos.writeObject(m);

oos.writeObject(r);

FileInputStream inos = new FileInputStream("C:\\Users\\gauta\\eclipse-workspace\\Serializations\\gsgds.ser");

ObjectInputStream ioos = new ObjectInputStream(inos);

Object obj=ioos.readObject();

if(obj instanceof Rat1) {

Rat1 ha =(Rat1) obj;

System.***out***.println(ha.a +"====>"+ha.b+"==>"+ha.c);

}

if(obj instanceof Dog1) {

Dog1 da =(Dog1) obj;

System.***out***.println(da.a +"====>"+da.b+"==>"+da.c);

}

if(obj instanceof Monkey) {

Monkey ma=(Monkey) obj;

System.***out***.println(ma.a +"====>"+ma.b+"==>"+ma.c);

}

**Object Graph:-**

An object Graph is a set of object which will be serialized Automatically. If the object which contain reference to them is serialized.

In other word we can say that we serialized any object if it contain any other object reference then JVM serialize the object as well as its object references.

For example:-

import java.io.\*;

class Dog123 implements Serializable{

Cat123 c =new Cat123();

}

class Cat123 implements Serializable{

Rat123 r =new Rat123();

}

class Rat123 implements Serializable{

int i=10;

}

public class objectgraph {

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

System.***out***.println("Serialization Started");

FileOutputStream fos = new FileOutputStream("Rarr.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

Dog123 d1 = new Dog123();

oos.writeObject(d1);

System.***out***.println("Serialization Ended");

FileInputStream ios = new FileInputStream("C:\\Users\\gauta\\eclipse-workspace\\Serializations\\Rarr.ser");

ObjectInputStream ioos = new ObjectInputStream(ios);

Dog123 d3 =(Dog123)ioos.readObject();

System.***out***.println(d3.c.r.i);

As we can see in the example, object reference are chained in other class so it gets serialized by chaining.

=> In the above example whenever we are serializing Dog object automatically Cat and Rat objects will be serialized because these are part of object graph of Dog object.

=> Among Dog, Cat, Rat if at least one object is not serializable then we will get runtime exception saying "NotSerializableException".

**Customized Serialization:-**

🡺 Now whenever we make any variable transient and then serialize it. The transient value didn’t get serialized due to transient keyword.

🡺So after deserialization, it didn’t get deserialized and we get null value for that transient variable to avoid that situation we use Customized Serialization. Let’s see how:-

We can implement Customized Serailization by following two method:-

1. private void writeObject(ObjectOutputStream os) throws Exception.

=> This method will be executed automatically by jvm at the time of serialization.

=> It is a callback method. Hence at the time of serialization if we want to perform any extra work we have to define that in this method only. (prepare encrypted password and write encrypted password

seperate to the file )

2. private void readObject(ObjectInputStream is) throws Exception.

=> This method will be executed automatically by JVM at the time of Deserialization.Hence at the time of Deserialization if we want to perform any extra activity we have to define that in this method only.

(read encrypted password , perform decryption and assign decrypted password to the Current object

password variable)

Before Customized Serialization:-

class Account implements Serializable{

int account =1234536;

transient int passw=103435;

}

public class CustomizedSerialization {

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

System.out.println("Before Serialization");

Account ac = new Account();

System.out.println(ac.account);

System.out.println(ac.passw);

System.out.println("serialization started");

FileOutputStream fos = new FileOutputStream("cust.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(ac);

FileInputStream ifos = new FileInputStream("cust.ser");

ObjectInputStream ioos = new ObjectInputStream(ifos);

Account act =(Account)ioos.readObject();

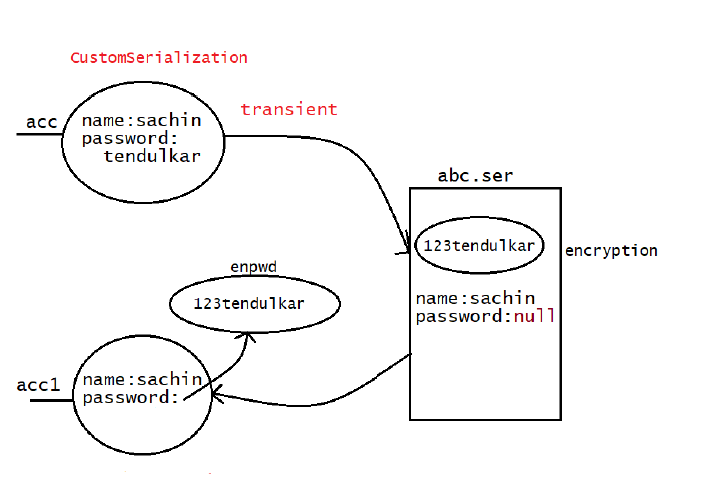
System.out.println(act.account+"====>"+act.passw);

}

}

🡺 In the above code, there is no customized serialization, so here password will be vanished. To avoid the loss of data from serialization in case of transient data we use customized serialization.

🡺To use Customized serialization, we need to use two method which is described above and need to take the help of encryption and decryption, let’s see how we will do it with the help of memory map diagram.



🡺As we can see in the diagram, there is name and password where password is transient data, to use customized serialization we need to use two method i.e.

* **Private void writeObject()**
* **Private void readObject()**

🡺 In the body of the writeObject we need to encrypt the password, encrypting is done by adding extra thing

from the password.

🡺In the body of the readObject(), we need to decrypt the Password by removing extra thing from the password let’s see how by using the code.

import java.io.\*;

class Account implements Serializable{

int account =1234536;

transient String passw="103435";

private void writeObject(ObjectOutputStream oos) throws Exception {

oos.defaultWriteObject();//all the properties written

String encPsw="123"+passw;

oos.writeObject(encPsw);//encrypted Password;

}

private void readObject(ObjectInputStream ioos) throws Exception {

ioos.defaultReadObject();

String enpwd =(String)ioos.readObject();//Decreypted Password

passw =enpwd.substring(3);

}

}

public class CustomizedSerialization {

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

System.out.println("Before Serialization");

Account ac = new Account();

System.out.println(ac.account);

System.out.println(ac.passw);

System.out.println("serialization started");

FileOutputStream fos = new FileOutputStream("cust.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(ac);

FileInputStream ifos = new FileInputStream("cust.ser");

ObjectInputStream ioos = new ObjectInputStream(ifos);

Account act =(Account)ioos.readObject();

System.out.println(act.account+"====>"+act.passw);

}

}

🡺As we can see in the program, password is being encrypted by adding extra thing in the password and it is decrypted by removing extra thing from the password.

🡺So this is the way to customized the serialization.

🡺 At the time of Account object serialization JVM will check is there any writeObject() method in Account class or not.

🡺 If it is not available then JVM is responsible to perform serialization(default serialization).

🡺 If Account class contains writeObject() method then JVM feels very happy and executes that Account class writeObject() method. The same rule is applicable for readObject() method also.

Example 2:-

import java.io.\*;

class Account implements Serializable{

String name="sachin";

transient String password="tendulkar";

transient int pin=4444;

private void writeObject(ObjectOutputStream oos)throws Exception{

oos.defaultWriteObject();//performing default Serialization

String epwd="123"+password;//performing encryption

int epin=1234+pin;//performing encryption

oos.writeObject(epwd);//write the encrypted data to file(abc.ser)

oos.writeInt(epin);//write the encrypted data to file(abc.ser)

}

private void readObject(ObjectInputStream ois)throws Exception{

ois.defaultReadObject();//performing default Serialization

String epwd=(String)ois.readObject();//performing decryption

int epin=ois.readInt();//performing decryption

password=epwd.substring(3);//writing the extra data to Object

pin=epin-1234;//writing the extra data to Object

}

}

public class Test {

public static void main(String[] args)throws

IOException,ClassNotFoundException{

Account acc=new Account();

System.out.println(acc.name +"=====> "+

acc.password+"======>"+acc.pin);

System.out.println("Serialization Started");

FileOutputStream fos= new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(acc);

System.out.println("Serialization ended");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("DeSerialization Started");

FileInputStream fis= new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

acc=(Account)ois.readObject();

System.out.println(acc.name +"=====> "+

acc.password+"========>"+acc.pin);

System.out.println("DeSerialization ended");

}

}

Output

sachin=====> tendulkar======>4444

Serialization Started

Serialization ended

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DeSerialization Started

sachin=====> tendulkar========>4444

DeSerialization ended

**Serialization with respect to Inhertiance.**

**Case 1:**

If parent class implements Serializable then automatically every child class by default implements Serializable. That is Serializable nature is inheriting from parent to child. Hence even though child class doesn't implements Serializable, we can serialize child class object if parent class implements serializable interface.

**Example:-**

import java.io.Serializable;

class Animal implements Serializable

{

int i =10;

}

class Dog extends Animal{

int j =20;

}

public class serialWithInheritance {

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

Dog d = new Dog();

System.out.println("serialisation started");

FileOutputStream fos = new FileOutputStream("serwitInhe.ser");

ObjectOutputStream oos = new ObjectOutputStream(fos);

oos.writeObject(d);

System.out.println("Serialisation Ended");

FileInputStream inos = new FileInputStream("serwitInhe.ser");

ObjectInputStream inoos = new ObjectInputStream(inos);

Dog da=(Dog)inoos.readObject();

System.out.println(da.i+"===>"+da.j);

}

}

**Output**

Serialization started

Serialization Ended

10====>20

🡺Even though Dog class does not implements Serializable interface explicitly but we can Serialize Dog object because its parent class Animal already implements Serializable interface.

**Note :**

Object class doesn't implement Serializable interface.

**Case 2:-**

🡺Suppose child class implements serialization and parent class doesn’t implements serialization.

🡺At that point of time, JVM will automatically discard the value of instance variable which is coming from non-serialized parents. Then instead of original value JVM will save the default value in the instance variable.

🡺At the time of Deserialization JVM checks whether any parent class is Non Serializable or not. If any parent class is Non Serializable JVM creates a separate object for every non Serializable parent and shares its instance variables to the current object.

🡺To create an object for non-serializable parent JVM always calls no arguments constructor(default constructor) of that non Serializable parent hence every non Serializable parent should compulsory contain no arguments constructor otherwise we will get runtime exception "InvalidClassException".

🡺If case of non-serializable parent class then just instance control flow will be performed and share it's instance variable to the current object.

EX:-

import java.io.IOException;

class Animal {

int i=10;

Animal(){

System.out.println("No arg Animal constructor");

}

}

class Dog extends Animal implements Serializable{

int j=20;

Dog(){

System.out.println("No arg Dog constructor");

}

}

public class Test {

public static void main(String[] args)throws

IOException,ClassNotFoundException{

Dog d=new Dog();

d.i=888;

d.j=999;

System.out.println("Serialization started");

FileOutputStream fos=new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(d);

System.out.println("Serialization ended");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("DeSerialization started");

FileInputStream fis=new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

Dog d1=(Dog)ois.readObject();

System.out.println(d1.i+"====> "+d1.j);

System.out.println("DeSerialization ended");

}

}

Output

No arg Animal constructor

No arg Dog constructor

Serialization started

Serialization ended

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DeSerialization started

No arg Animal constructor

10====> 999

DeSerialization ended

**Externalization : ( 1.1 v )**

1. In default serialization every thing takes care by JVM and programmer doesn't have any control.

2. In serialization total object will be saved always and it is not possible to save part of the object , which creates performance problems at certain point.

3. To overcome these problems we should go for externalization where every thing takes care by programmer and JVM doesn't have any control.

4. The main advantage of externalization over serialization is we can save either total object or part of the object based on our requirement.

5. To provide Externalizable ability for any object compulsory the corresponding class should implements externalizable interface.

6. Externalizable interface is child interface of serializable interface. Externalizable interface defines 2 methods :

**1. writeExternal(ObjectOutput out ) throws IOException**

**2. readExternal(ObjectInput in) throws IOException,ClassNotFoundException**

**public void writeExternal(ObjectOutput out) throws IOException**

This method will be executed automaticcay at the time of Serialization with in this method , we have to write code to save required variables to the file.public void readExternal(ObjectInput in) throws IOException,ClassNotFoundException This method will be executed automatically at the time of deserialization with in this method , we have to write code to save read required variable from file and assign to the current object. At the time of deserialization JVM will create a seperate new object by executing public no-arg constructor on that object JVM will call readExternal() method. Every Externalizable class should compusory contains public no-arg constructor otherwise we will get RuntimeExcepion saying "InvaidClassException" .

eg#1.

import java.io.Serializable;

import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

import java.io.FileInputStream;

import java.io.ObjectInputStream;

import java.io.IOException;

import java.io.Externalizable;

import java.io.ObjectOutput;

import java.io.ObjectInput;

class ExternalizableDemo implements Externalizable{

String i;

int j;

int k;

ExternalizableDemo(String i,int j,int k){

this.i=i;

this.j=j;

this.k=k;

}

public ExternalizableDemo(){

System.out.println("Zero arg constructor");

}

//Performing Serialization as per our requirement

public void writeExternal(ObjectOutput out) throws IOException{

System.out.println("call back method used while Serialization");

out.writeObject(i);

out.writeInt(j);

}

//Performing DeSerialization as per our requirement

public void readExternal(ObjectInput in) throws

IOException,ClassNotFoundException{

System.out.println("call back method used while DeSerialization");

i=(String)in.readObject();

j=in.readInt();

}

}

public class Test {

public static void main(String[] args)throws

IOException,ClassNotFoundException{

ExternalizableDemo d=new ExternalizableDemo("nitin",100,200);

System.out.println("Serialization started");

FileOutputStream fos=new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(d);

System.out.println("Serialization ended");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("DeSerialization started");

FileInputStream fis=new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

d=(ExternalizableDemo)ois.readObject();

System.out.println(d.i+"======>"+d.j+"======>"+d.k);

System.out.println("DeSerialization ended");

}

}

Output

Serialization started

call back method used while Serialization

Serialization ended

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

DeSerialization started

Zero arg constructor

call back method used while DeSerialization

nitin======>100======>0

DeSerialization ended

1. If the class implements Externalizable interface then only part of the object will be saved in the case output is public no-arg constructor nitin---- 10 ----- 0

2. If the class implements Serializable interface then the output is nitin --- 10--- 20

3. In externalization transient keyword won't play any role, hence transient keyword not required.

Difference b/w Serialization and Externalization

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

Serialization

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

1. It is meant for default Serialization

2. Here every thing takes care by JVM and programmer doesn't have any control

doesn't have any control.

3. Here total object will be saved always and it is not possible to save part of

the object.

4. Serialization is the best choice if we want to save total object to the file.

5. relatively performence is low.

6. Serializable interface doesn't contain any method

7. It is a marker interface.

8. Serializable class not required to contains public no-arg constructor.

9. transient keyword play role in serialization

Externalization

1. It is meant for Customized Serialization

2. Here every thing takes care by programmer and JVM does not have any control.

3. Here based on our requirement we can save either total object or part of the object.

4. Externalization is the best choice if we want to save part of the object.

5. relatively performence is high

6. Externalizable interface contains 2 methods :

1. writeExternal()

2. readExternal()

7. It is not a marker interface.

8. Externalizable class should compulsory contains public no-arg constructor

otherwise we will get RuntimeException saying "InvalidClassException"

9. transient keyword don't play any role in Externalization.

shallow copy, deep copy

**Cloneable object**

**serialVersionUID**

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

🡺 To perform Serialization & Deserialization internally JVM will use a unique identifier,which is nothing but serialVersionUID .

🡺 At the time of serialization JVM will save serialVersionUID with object.

🡺 At the time of Deserialization JVM will compare serialVersionUID and if it is matched then only object will be Deserialized otherwise we will get RuntimeException saying "InvalidClassException".

The process in depending on default serialVersionUID are :-

1. After Serializing object if we change the .class file then we can't perform deserialization because of mismatch in serialVersionUID of local class and serialized object in this case at the time of Deserialization

we will get RuntimeException saying in "InvalidClassException".

2. Both sender and receiver should use the same version of JVM if there any

incompatability in JVM versions then receive anable to deserializable because of different serialVersionUID , in this case receiver will get RuntimeException saying "InvalidClassException".

3. To generate serialVersionUID internally JVM will use complexAlgorithm which may create performence problems.

**Serialization**

class Dog implements Serializable{

public static final long serialVersionUID = 1L;

int i=10;

int j=20;

}

FileOutputStream fos= new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(d1);

System.out.println("Serialization ended");

**DeSerialization**

class Dog implements Serializable{

public static final long serialVersionUID = 1L;

int i=10;

int j=20;

}

System.out.println("Deserialization started");

FileInputStream fis=new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

Dog d2=(Dog) ois.readObject();

System.out.println("Deserialization ended");

We can solve above problems by configuring our own serialVersionUID .

eg#1.

import java.io.Serializable;

public class Dog implements Serializable {

private static final long serialVersionUID=1L;

int i=10;

int j=20;

}

import java.io.\*;

public class Sender {

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

Dog d=new Dog();

FileOutputStream fos=new FileOutputStream("abc.ser");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(d);

}

}

import java.io.\*;

public class ReceiverApp {

public static void main(String[] args) throws

IOException,ClassNotFoundException{

FileInputStream fis=new FileInputStream("abc.ser");

ObjectInputStream ois=new ObjectInputStream(fis);

Dog d2=(Dog) ois.readObject();

System.out.println(d2.i+"=====>"+d2.j);

}

}

🡺 In the above program after serialization even though if we perform any change to Dog.class file we can deserialize object.

🡺We can configure our own serialVersionUID both sender and receiver not required to maintain the same JVM versions.

**Note : some IDE's generate explicit serialVersionUID**

**Clone method()**

1. The process of creating exactly duplicate object is called cloning.

2. The main objective of cloning is to maintain backup purposes.(i.e., if something goes wrong we can recover the situation by using backup copy.)

3. We can perform cloning by using clone() method of Object class.

**Signature**

**protected native object clone() throws CloneNotSupportedException;**

**Example:-**

public class clone1 implements Cloneable {

int i=10;

int j= 20;

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

clone1 t1 =new clone1();

clone1 t2 = (clone1)t1.clone();

t2.i=100;

t2.j=200;

System.out.println(t2.i+"==>"+t2.j);

System.out.println(t1.i+"==>"+t1.j);

}

}

Output

10🡺20

10🡺20

**KeyPoints about Cloneable interfaces**

**===================================**

🡺 We can perform cloning only for Cloneable objects.

🡺 An object is said to be Cloneable if and only if the corresponding class implements Cloneable interface.

🡺 Cloneable interface present in java.lang package and does not contain any methods. It is a marker interface where the required ability will be provided automatically by the JVM.

🡺 If we are trying to perform cloning on non-clonable objects then we will get RuntimeException saying "CloneNotSupportedException".

Ex:- package clone12;

class Cat{

int i;

Cat(int i){

this.i=i;

}

}

class Dog implements Cloneable{

Cat cat;

int j;

Dog(Cat cat, int j){

this.cat=cat;

this.j =j;

}

@Override**//clone is protect method so we need to override it.**

public Object clone()throws CloneNotSupportedException{

return super.clone();

}

}

public class clone2 {

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

Cat c = new Cat(10);

Dog d = new Dog(c,20);

Dog d1 =(Dog)d.clone();

d1.cat.i=100;

d1.j=200;

System.out.println(d.cat.i +"==>"+d.j);

System.out.println(d1.cat.i +"==>"+d1.j);

}

}

**Above cloning is called Shallow Cloning**

**Shallow Cloning**

**🡺** A **shallow copy** creates a new instance of the object and copies all the fields of the original object to the new instance.

🡺 However, if the object contains references to other objects, the references are copied, not the actual objects. This means that both the original and the copied objects share the same references to the nested objects.

🡺In Deep cloning, if main object contain any reference variable then the corresponding Object copy will also be created in cloned object.

🡺To overcome this problem we should go for Deep cloning.

**Deep Cloning**

**🡺**A **deep copy** creates a new instance of the object and also creates new instances of any nested objects. This means that the original and the copied objects are completely independent of each other.

🡺The process of creating exactly independent duplicate object(including contained objects also) is called deep cloning.

🡺In Deep cloning , if main object contain any reference variable then the corresponding Object copy will also be created in cloned object.

🡺Object class clone( ) method meant for Shallow Cloning , if we want Deep cloning then the programmer is responsible to implement by overriding clone( ) method.

package clone12;

class Cat1 implements Cloneable{

int i;

Cat1(int i){

this.i=i;

}

}

class Dog2 implements Cloneable{

Cat1 cat;

int j;

Dog2(Cat1 cat, int j){

this.cat=cat;

this.j =j;

}

@Override//clone is protect method so we need to override it.

public Object clone()throws CloneNotSupportedException{

Cat1 c1 = new Cat1(cat.i); // cat object is also created in deep cloning

Dog2 d = new Dog2(c1,c1.i);// Dog object is also created in deep cloning

return d ;

}

}

public class DeepCopy {

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

Cat c = new Cat(10);

Dog d = new Dog(c,20);

Dog d1 =(Dog)d.clone();

d1.cat.i=100;

d1.j=200;

System.out.println(d.cat.i +"==>"+d.j);

System.out.println(d1.cat.i +"==>"+d1.j);

}

}

🡺As we see in the program, clone method is manually written so we need to create each object whether it is a nested object or copied object.

**Singleton classes :**

For any java class if we are allow to create only one object, such type of class is said to be singleton class.

**Example:**

1) Runtime class

2) ActionServlet

3) ServiceLocator

4) BusinessDelegate

eg#1

Runtime r1=Runtime.getRuntime();//getRuntime() method is a factory method

Runtime r2=Runtime.getRuntime();

Runtime r3=Runtime.getRuntime();

.................................................

.................................................

System.out.println(r1==r2);//true

System.out.println(r1==r3);//true

**Advantage of Singleton class:-**

If the requirement is same then instead of creating a separate object for every person we will create only one object and we can share that object for every required person we can achieve this by using singleton classes.

That is the main advantages of singleton classes are Performance will be improved and memory utilization will be improved.

**Creation of our own singleton classes:**

We can create our own singleton classes for this we have to use private constructor, static variable and factory method.

To Create a Single ton class we need to follow these rule let’s see through example:-

Line no 1🡺 class Demo{

Line no 2🡺 private static Demo d =null;

Line no 3🡺 private Demo() {

Line no 4🡺 }

Line no 5🡺 public static Demo getDemo() {

Line no 6🡺 if (d==null) {

Line no 7🡺 d=new Demo();

Line no 8🡺 return d;

Line no 9🡺 }

Line no 10🡺 else

Line no 11🡺 return d;

Line no 12🡺 }

Line no 13🡺 }

Line no 14🡺 public class single1 {

Line no 15🡺 public static void main(String[] args) {

Line no 16🡺 Demo d1 = Demo.getDemo();

Line no 17🡺 Demo d2 = Demo.getDemo();

Line no 18🡺 System.out.println(d1==d2);

Line no 19🡺 }

Line no 20🡺 }

🡺First we need to class which should have

1.private static dataset or variable which have same class return type,

2.private constructor

3.public static method which will create the object for the class.

🡺As we can see in the above code, there is demo class which has private reference variable of same class initialised as null, private constructor and getdemo() method which is of static method of return type demo.

🡺In the getDemo() method, we will first check the variable if it is null or not, if it is null then we will create the object and stores in that static variable and return it.

🡺If the variable is not null, then we will return that variable.

🡺In the single1 class, we can create the object of the class as constructor is private so we can only call static method which is creating the object so simply what we do is we create the reference of the demo class and in that reference we store the value of **getdemo()** method as it is created object.

🡺So whenever we call **getDemo**() method, it simply pass the object which is already created or the new object. Or if we again call getDemo() method , it will simply return the created object, it will not create the new one.

**Factory method:**

By using class name if we are calling a method and that method returns the same class object such type of method is called factory method.

We can create singleton class as describe above but we can create doubleton or tripleton class by using this way

class Demo2{

private static Demo2 d1 = null;

private static Demo2 d2 =null;

private Demo2() {

}

public static Demo2 getDemo2() {

if(d1==null) {

d1 =new Demo2();

return d1;

}

else if(d2==null) {

d2 = new Demo2();

return d2;

}

else {

if(Math.random()<0.5) {

return d1;

}

else {

return d2;

}

}

}

}

public class double1 {

public static void main(String[] args) {

System.out.println(Demo2.getDemo2().hashCode());

System.out.println(Demo2.getDemo2().hashCode());

System.out.println(Demo2.getDemo2().hashCode());

System.out.println(Demo2.getDemo2().hashCode());

}

}

**Different ways of Creating an Object**

1. using new Operator

**Test t=new Test();**

2. using newInstance()

**Class.forName("com.abc.main.Test").newInstance()**

3. using clone()

**Test t2=(Test)t1.clone();**

4. using factorymethods

**Runtime r=Runtime.getRuntime();**

**DateFormat df=DataFormat.getInstance();**

5. using Serialization and DeSerialization

**FileInputStream fis=new FileInputStream("abc.ser");**

**ObjectInputStream ois=new ObjectInputStream(fis);**

**Test t=(Test)ois.readObject();**

**2.** **Using newInstance()**

🡺newInstance() is a method which is generally used to create object dynamically. Dynamically mean during runtime, suppose we don’t know the name of the class, At that time, how we will create the object when we pass the class name using some argument during runtime.

Suppose we have placed “java.lang.String” in one of the argument then how we will create the object from that argument

**Arg[0]=** “java.lang.String”

At that we will use one factory method i.e.

**forName()**

This is a method which extract the class name from the argument through which we can create the object.

Such as

Strring data = Class.forName(arg[0]).newInstance();

It will simply create the object of the class.

class Student {

static {

System.out.println("Student class loading");

}

public Student() {

System.out.println("Student object get crated");

}

}

public class newInst {

public static void main(String[] args)

throws InstantiationException, IllegalAccessException, ClassNotFoundException {

Student std = (Student) (Class.forName(args[0]).newInstance());

}

}

**Difference between ClassNotFoundException & NoClassDefFoundError :**

1. For hard coded class names at Runtime in the corresponding .class files not available we will get **NoClassDefFoundError** , which is unchecked

Test t = new Test( );

In Runtime Test.class file is not available then we will get "NoClassDefFoundError"

2. For Dynamically provided class names at Runtime , If the corresponding .class files is not available then we will get the RuntimeException saying "**ClassNotFoundException**".

Ex : Object o=Class.forname("Test").newInstance( );

At Runtime if Test.class file not available then we will get the "ClassNotFoundException" , which is checked exception.

**Difference between instanceof() and isInstance( ) :**

**Instanceof()**

instanceof an operator which can be used to check whether the given object is particular type or not We know at the type at beginning it is available.

eg: String s = new String("sachin");

System.out.println(s instanceof Object );//true

//If we know the type at the beginning only.

**isInstance( )**

isInstance( ) is a method , present in class Class , we can use isInstance( ) method to checked whether the given object is particular type or not We don't know at the type at beginning it is available Dynamically at

Runtime.

class Test {

public static void main(String[] args) {

Test t = new Test( ) ;

System.out.println(Class.forName(args[0]).isInstance(t));////arg[0] --- We

don't know the type at beginning

}

}

java Test Test //true

java Test String //false

java Test Object //true

**SQL(Structure Query Language)**

It is a declarative language that means we just need to declare SQL to fetch the Code.

There are five type of SQL :-

* DDL
* DML
* DCL
* TCL
* DQL

DDL –DDL stands for data definition Language. It generally use for creation of table.(Create,Drop,Alter,Truncate,comment,Rename).

DML-DML stands for data definition Language. It generally used to manipulate the data in the table.(Insert,update,delete,lock,call,Explain).

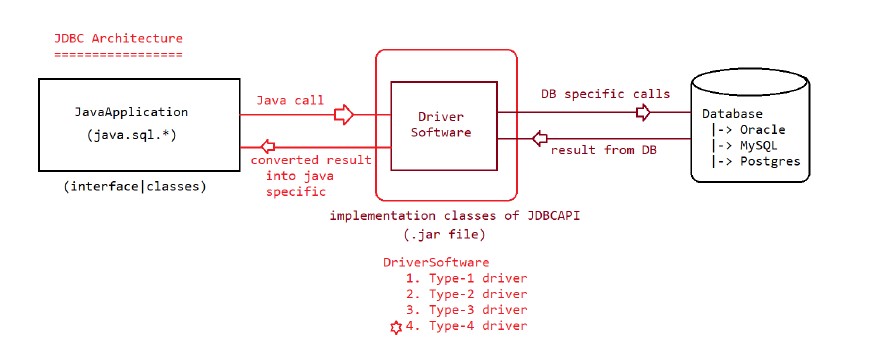
DCL-DCL stands for data control language. It is generally used to provide the control to table like **Grant,Revoke,**

TCL-TCL stands for Transaction control Language. It work with transcation like **Commit,Rollback,Savepoint etc.**

DQL- DQL stands for data Query langage like (select).

**JDBC(Java Database Connectivity)**

**JDBC** stands for Java Database Connectivity. It is kind of API which is provided by sunmicrosystem to connect to different Database Vendor. Let’s understand through pictorial representation and then we will try to implement it.



****

**🡺**As we can see in the diagram, JDBC is API which is provided by sunmicrosystem for connecting java application with the database(ORACLE,MYSQL,POSTGRESQL)

🡺JDBC is nothing but package of interface which present in rt.jar file(runtime jar file). Simply sunmicrosystem what has done that it has created the interface package and ask the different database vendor to provide the implementation of that jar file.

🡺Now different database vendor has provide the implementation of the JDBC api by providing jar file of it’s database as they know their database more efficiently so it is the responsibility of the database vendors to provide the implementation of jdbc api.

🡺 Now suppose we have rt.jar and db.jar , now the question arise how we are going to keep the both jar in java file . The answer is Path Environment variable and class path jar.

Path Environment variable – It is environmental variable which is associated with operating system, this environmental variable is used to inform the location of .exe files of the software which needs to run from command prompt.

Set path =……./bin=>.exefile

ClassPath:-It is an environmental variable associated with “java program”. Java Developer uses this environmental variable and inform the jdk software to search for the required .class files.(normally used in jdbc,servlet,hibernate,springMVC)

Set classPath ==\*\*\*\*\*.jar

|=>.class files

**JDBC Archietrure**

Standard Steps followed for developing JDBC(JDBC4.X) Application

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

1. Load and register the Driver

2. Establish the Connection b/w java application and database

3. Create a Statement Object

4. Send and execute the Query

5. Process the result from ResultSet

6. Close the Connection

Step1:

**1. Load and register the Driver**

A third party db vendor class which implements java.sql.Driver(I) is called as "Driver".

This class Object we need to create and register it with JRE to set up JDBC environment to run jdbc applications.

**Note:**

public class com.mysql.cj.jdbc.Driver extends com.mysql.cj.jdbc.NonRegisteringDriver implements java.sql.Driver

{

public com.mysql.cj.jdbc.Driver() throws java.sql.SQLException;

static {};

}

In MySQL Jar, Driver class is implementing java.sql.Driver, so Driver class Object should be created and it should be registered to set up the JDBC environment inside JRE.

**Driver driver = new Driver();**

**DriverManager.registerDriver(driver);**//It will register the driver

**String url = "jdbc:mysql://localhost:3306/ineuron";**

**String username = "root";**

**String password = "Gaurav45";**

**Connection connection = DriverManager.getConnection(url, username, password);**

**2. Establish the Connection b/w java application and database**

public static Connection getConnection(String url, String username,String password) throws SQLException;

public static Connection getConnection(String url, Properties) throws SQLException;

public static Connection getConnection(String url) throws SQLException;

The below creates the Object of Connection interface.

**Connection connection = DriverManager.getConnection(url,username,password);**

getConnection(url,username,password) created an object of class which implements Connection(I)

that class object is collected by Connection(I).

This feature in java refers to

a. Abstraction(hiding internal services)

b. polymorphism(making code run in 1:M forms)

URL SYNTAX:: <mainprotocol>:<subprotocol>:<subname>

**String url = "jdbc:mysql://localhost:3306/ineuron";**

**String username = "root";**

**String password = "Gaurav45";**

**Connection connection = DriverManager.getConnection(url, username, password);**

**3. Create a Statement Object**

public abstract Statement createStatement() throws SQLException;

public abstract Statement createStatement(int,int) throws SQLException;

public abstract Statement createStatement(int,int,int) throws SQLException;

**Statement statement = connection.createStatement();**

**4. Send and execute the Query**

Query

=====

From DB administrator perspective queries are classified into 5 types

1. DDL (Create table,alter table,drop table,..)

2. DML(Insert,update,delete)

3. DQL(select)

4. DCL(alter password,grant access)

5. TCL(commit,rollback,savepoint)

According to java developer perspective, we catergorise queires into 2 types

a. Select Query

b. NonSelect Query

Methods for executing the Query are

a. executeQuery() => for select query we use this method.

b. executeUpdate() => for insert,update and delete query we use this method.

c. execute() => for both select and non-select query we use this method

public abstract ResultSet executeQuery(String sqlSelectQuery) throws SQLException;

String sqlSelectQuery ="select sid,sname,sage,saddr from Student";

ResultSet resultSet = statement.executeQuery(sqlSelectQuery);

**5. Process the result from ResultSet**

public abstract boolean next() throws java.sql.SQLException;

|=> To check whether next Record

is available or not

returns true if available

otherwise returns false.

System.out.println("SID\tSNAME\tSAGE\tSADDR");

while(resultSet.next()){

Integer id = resultSet.getInt(1);

String name = resultSet.getString(2);

Integer age = resultSet.getInt(3);

String team = resultSet.getString(4);

System.out.println(id+"\t"+name+"\t"+age+"\t"+team);

}

**6. Close the Connection**

**Ex:-**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**import com.mysql.cj.jdbc.Driver;**

**public class TestApp {**

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

**Driver driver = new Driver();// Creating driver object for MySQLDB**

**DriverManager.registerDriver(driver);**

**System.out.println("Driver registered succesfully");**

// Step2: Establish the connection b/w java and Database

// JDBC URL SYNTAX:: <mainprotocol>:<subprotocol>:<subname>

**String url = "jdbc:mysql://localhost:3306/ineuron";**

**String username = "root";**

**String password = "Gaurav45";**

**Connection connection = DriverManager.getConnection(url, username, password);**

**System.out.println("Connection object is created:: " + connection);**

**Statement statement = connection.createStatement();**

**System.out.println("Statement object is created:: " + statement);**

//Sending and execute the Query

**String sqlSelectQuery ="select sid,sname,sage,saddr from Student";**

**ResultSet resultSet = statement.executeQuery(sqlSelectQuery);**

**System.out.println("ResultSet object is created:: " + resultSet);**

//Process the result from ResultSet

**System.out.println("SID\tSNAME\tSAGE\tSADDR");**

**while(resultSet.next()){**

**Integer id = resultSet.getInt(1);**

**String name = resultSet.getString(2);**

**Integer age = resultSet.getInt(3);**

**String team = resultSet.getString(4);**

**System.out.println(id+"\t"+name+"\t"+age+"\t"+team);**

**}**

**//Close the Connection**

**connection.close();**

**System.out.println("Closing the connection...");**

**}**

**}**

**Output:-**

**Driver registered succesfully**

**Connection object is created:: com.mysql.cj.jdbc.ConnectionImpl@58ea606c**

**Statement object is created:: com.mysql.cj.jdbc.StatementImpl@68267da0**

**ResultSet object is created:: com.mysql.cj.jdbc.result.ResultSetImpl@61d6015a**

**SID SNAME SAGE SADDR**

**1 GAURAV 25 Barkakana**

**2 Rahul 45 Sasaram**

**3 Ram 34 Ramgarh**

**Closing the connection...**

**\*\*\*From JDBC4.x onwards loading and registering would happen automatically depending upon the jar added in the classPath location of the project.**

**Note:**

**1.JVM will search for the jar in the classpath**

**2.It will open the jar, move to META-INF folder**

**3. It will open services folder**

**4.It will search for java.sql.driver file**

**5.Whatever value which is present inside Driver file that would be loaded automatically using Class.forName(value).**

This is feature is called Auto-Loading.

Industry Code

**public class IdustryWay {**

**public static void main(String[] args) {**

**String url = "jdbc:mysql://localhost:3306/ineuron";**

**String username = "root";**

**String password = "Gaurav45";**

**try {**

**Connection connection = DriverManager.getConnection(url, username, password);**

**if (connection != null) {**

**Statement statement = connection.createStatement();**

**if (statement != null) {**

**String sqlSelectQuery = "select sid,sname,sage,saddr from Student";**

**ResultSet resultSet = statement.executeQuery(sqlSelectQuery);**

**while (resultSet.next()) {**

**Integer id = resultSet.getInt(1);**

**String name = resultSet.getString(2);**

**Integer age = resultSet.getInt(3);**

**String team = resultSet.getString(4);**

**System.out.println(id + "\t" + name + "\t" + age + "\t" + team);**

**}**

**connection.close();**

**System.out.println("Closing the connection...");**

**}**

**}**

**} catch (SQLException se) {**

**se.printStackTrace();**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

To resolve the problem of the above approach we use a inbuilt class called "String".

**int sage = scanner.nextInt();**

**String sname = scanner.next();**

**String saddr = scanner.next();**

**public static String format(String format, Object... args) {**

**return new Formatter().format(format, args).toString();**

**}**

**Note:-**

**String use format specifier as '%s'**

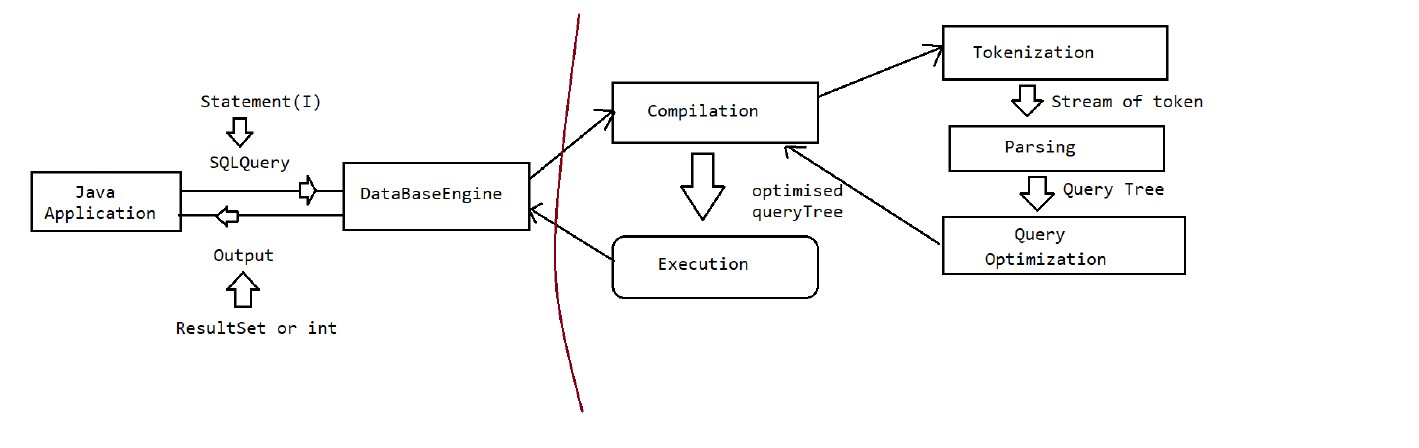
**int use format specifier as %d**

**flaot use format specifier as %f**

**String query =String.format( "insert into student(`sname`,`sage`,`saddr`) values**

**('%s',%d,'%s')",sname,sage,saddr );**

**Working of JDBC**



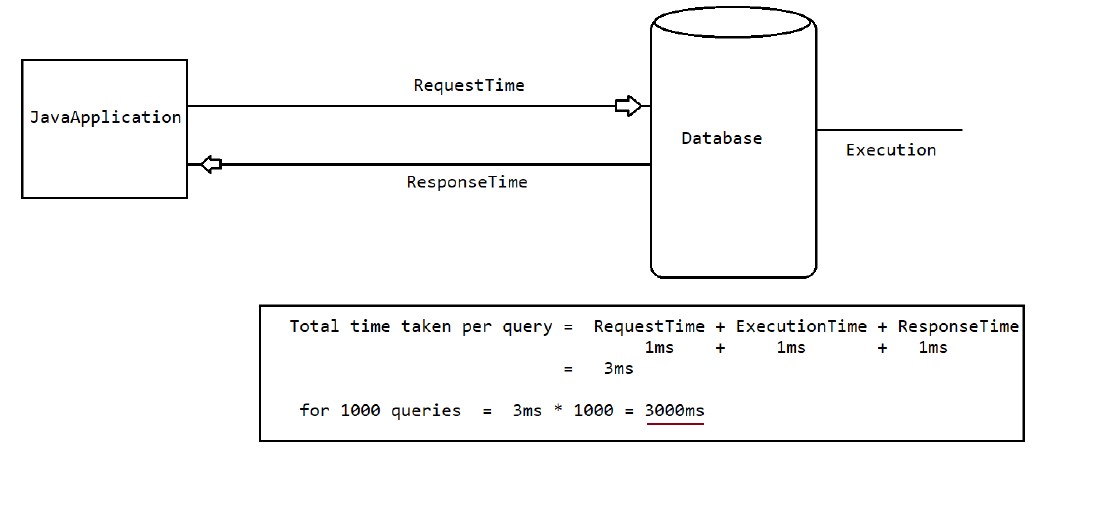
🡺 From the above diagram we can see what happen internally when we try to execute SQL query in JDBC

These are steps which is followed by JDBC to execute the SQL Query:-

1. **Java Application(Database driver loaded in it)**
2. **DataBaseEngine**
3. **Compilation(Tokenization,Parsing,Query Optimization)**
4. **Execution**

After the execution it will again go back to the Database engine and performs the operation which is specified in the query.

And Finally database Engine gives the result to the driver depends upon the output of the query in the form of (ResultSet or Int).



In the above fig as we can see that if there is any java application which is using database to retrieve the data these are the steps first it will request the data then compilation will happen then execution will happen then it will give the response time for the query.

1. Request
2. Compilation
3. Execution
4. Response

Now suppose if each steps takes 1ms of time to process the query then it will take approximately 4 ms to execute the one query.

i.e.

**Total time taken = RequestTime(1ms)+CompilationTime(1ms)+ExecutionTime(1ms)+Responsetime(1ms)=4ms**

Now suppose there is 1000 query to execute than it will take approximately =1000\*4ms=4000ms

Let’s understand will the Example:-

Suppose we are using the IRCTC to search the train then what will the query here:-

**Select \* from irctc where departure =’XXXXX’ and destination=’XXXXX’;**

Now suppose if the same query is executed by 10000 user then each time JDBC try to fetch the value whenever user search the train from the website. As there are so many user present in the website site response get delayed so what will the solution here.

From the above query we can see that if any user search for the train the query will be same but input will only be different. It mean any user want to go Delhi from Agra and other user want to go Banglore from Ranchi.

**So What we can do here is now suppose what if we load all the value for the selected query in database engine initially and whenever anybody search for the train query will directly fetch from the database Engine instead of going to the database and it will help reduce the compilation and execution time.**

**To do this instead of using Statement Object we will use *Prepared Statement* (Pre-Compiled Statement)**.

Signature

public PreparedStatement prepareStatement(String sqlQuery) throws SQLException

//Establish the Connection

**Connection con= DriverManager.getConnection(url,username,password);**

//Creating a preCompiled query which is used at the runtime to execute with the value

**String sqlSelectQuery =’select sid,dname,sage,saddr from student where sid=?’;**

**PreparedStatement pstmt =con.prepareStatement(sqlSelectQuery);**

At this line,sqlquery will be sent to database,Database engine will compile the query and stores in database. That precompiled query will be sent to the java application in the form of “PreparedStatement” object.Hence PreparedStatement Object is called as Pre-compiled Statement.

//Execute the Query by settling the input for preCompiledQuery

**Integer sid=10;**

**pstmt.setInt(1,sid)**

**ResultSet resultSet =pstmt.executeQuery();**

Whenever we execute methods, databaseengine will not compile query once again and it will directly execute that query, so that performance will be improved.

**Note:**

**String sqlQuery= insert into student(`sid`,`sname`,`sage`,`saddres`)**

**values(?,?,?,?);**

**PreparedStatement pstmt = con.prepareStatement(sqlQuery);**

**pstmt.setInt(1,10);**

**pstmt.setString(2,"sachin");**

**pstmt.setInt(3,45);**

**pstmt.setString(4,"MI");**

**int rowCount = pstmt.executeUpdate();**

KeyPoints of methods

Selectquery =>executeQuery()

nonSelectQuery=> executeUpdate()

both select and nonSelect Quert =execute();

First of all we will try to write the utility code which is generally used again and again like creating connection and closing the connection so we will simply write the class which creates the connection.

Inside the class there is a method which public static and it’s return type is of connection type. We are making the method static so that we don’t need to create the object of it we can directly call it from the class name.

We will create the constructor private so that no one can create the object of it from outside the class.

**public class Connectionjdbc {**

**private Connectionjdbc() {**

**}**

**public static Connection getJdbcConnection() throws SQLException {**

**Connection connection = null;**

**String url = "jdbc:mysql://localhost:3306/ineuron";**

**String id = "root";**

**String pass = "Gaurav45";**

**try {**

**connection = DriverManager.getConnection(url, id, pass);**

**if(connection!=null)**

**return connection;**

**}**

**catch (SQLException se) {**

**se.printStackTrace();**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**return connection;**

**}**

**public static void closeConnection(ResultSet rs, Statement st,Connection con) throws SQLException {**

**if(st!=null)**

**st.close();**

**if(con!=null)**

**con.close();**

**if(rs!=null)**

**rs.close();**

**}**

**}**

Now let’s how we are going to create the preapare statement for fetching the query by using the prepare statement.

**package in.neuron.dynamicinput;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.util.Scanner;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class selectApp {**

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

**Connection connection = null;**

**PreparedStatement pstmt = null;**

**ResultSet resultSet = null;**

**Scanner sc = new Scanner(System.in);**

**System.out.println("enter the sid");**

**int sid = sc.nextInt();**

**String sqlSelctQuery = "select sid,sname,sage,saddr " + "from student where sid =? ";**

**try {**

**connection = Connectionjdbc.getJdbcConnection();**

**if (connection != null) {**

**pstmt = connection.prepareStatement(sqlSelctQuery);**

**if (pstmt != null) {**

**pstmt.setInt(1, sid);**

**resultSet = pstmt.executeQuery();**

**if (resultSet != null) {**

**while (resultSet.next()) {**

**Integer id = resultSet.getInt(1);**

**String name = resultSet.getString(2);**

**Integer age = resultSet.getInt(3);**

**String addr = resultSet.getString(4);**

**System.out.println(id + "\t" + name + "\t" + age + "\t" + addr);**

**}**

**}**

**else {**

**System.out.println("Record not found");**

**}**

**}**

**}**

**} catch (SQLException se) {**

**se.printStackTrace();**

**} catch (Exception e) {**

**e.printStackTrace();**

**} finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**if (sc != null)**

**sc.close();**

**}**

**}**

**}**

Now let’s see how we are going to insert the values using prepare statement.

**package in.neuron.dynamicinput;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**import java.util.Scanner;**

**import com.mysql.cj.jdbc.JdbcConnection;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class Insertapp {**

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

**Connection connection=null;**

**PreparedStatement pstmt=null;**

**Scanner sc =new Scanner(System.in);**

**System.out.println("enter the age");**

**int sage =sc.nextInt();**

**System.out.println("Enter the sname");**

**String sname =sc.next();**

**System.out.println("enter the saddress");**

**String saddr =sc.next();**

**String sqlInsertQuery="Insert into student(sname,sage,saddr)"**

**+ "values(?,?,?)";**

**try {**

**connection=Connectionjdbc.getJdbcConnection();**

**if(connection!=null) {**

**pstmt=connection.prepareStatement(sqlInsertQuery);**

**if(pstmt!=null) {**

**pstmt.setString(1 ,sname);**

**pstmt.setInt(2, sage);**

**pstmt.setString(3, saddr);**

**int rowaffected=pstmt.executeUpdate();**

**System.out.println("No.of row affected "+rowaffected);**

**}**

**}**

**} catch (SQLException se) {**

**se.printStackTrace();**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**if(sc!=null)**

**sc.close();**

**}**

**}**

**}**

**SQL Injection**

SQL injection is a code technique that might destroy your database. It is one of the most common web hacking technique which is generally used to pass some query to get authentication.

Statement object is generally used for SQL injection, if any developer has used statement object for retrieving the query from database. Suppose we have table USER in which there is two column username and password like

**USER**

**Username userpsd**

**Rohit sharma**

**Sachin tendulakar**

Now suppose we are using this query to count the row

Select count(\*) from user where username =”Sachin” and userpsd =”Tendulkar”;

It will give the output ==1

But now suppose if we use same query like this way:-

Select count(\*) from user where username =”Sachin” --and userpsd =”Tendulkar”;

It will also give the output =1

Here “–“ is use to comment the query in the sql;

Let’s see why it also get authenticated.

e.g.

**String query = Select count(\*) from user where username =”Sachin” --and userpsd =”Tendulkar”;**

**Statement** **stmt =con.createStatement();**

**Resultset rs =stmt.executeQuery(query);**

Inside the DB it will comment the second half of the query and just read the first half the query i.e.

Select count(\*) from user where username =”Sachin”;

It will validate first and gives the output 1. This is one of the drawback of statement object in java. But this problem can be solve by using prepare statement instead of statement object.

if we use PreparedStatement Object to send the Query, then the problem of SQLInjection will not happen.

eg: String query = "select count(\*) from users where username =? and upwd =?";

PreparedStatement pstmt = con.prepareStatement(query);

pstmt.setString(1,"sachin'--");

pstmt.setString(2,"tendulkar");

ResultSet resultSet =pstmt.executeQuery();

|

| for compilation using PreparedStatement

|

DB: select count(\*) from users where username =? and upwd =?;

||

select count(\*) from users where username ='sachin'--' and upwd

='tendulkar';

|

count(\*) => 0 (validation not succesfull so no authentication)

**Note: In real time database used in production envrionment is "Oracle", only during development phase we use "MySQL" database. In MySQLDatabase, we can't perform "SQLInjection" through comments,it**

**happens only in "OracleDatabase".**

eg:

select \* from users where userid = 1; (1 record will be pulled)

select \* from users where userid= 1 or 1=1;(All records in the table will be pulled).

**How to handle Date object in Database?**

Handling Date Values For Database Operations

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

=> Sometimes as the part of programing requirement,we have to insert and retrieve Date like DOB,DOJ,DOM,DOP...wrt database.

=> It is not recommended to maintain date values in the form of String, because comparisons will become difficult.

In Java we have two Date classes

1. java.util.Date

2. java.sql.Date

=> java.sql.Date is the child class of java.util.Date.

=> java.sql.Date is specially designed class for handling Date values wrt database. Otherthan database operations,if we want to represent Date in our java program then we should go for java.util.Date.

=> java.util.Date can represent both Date and Time where as java.sql.Date represents only Date but not time.

**public static void main(String[] args) {**

**Date dateu =new java.util.Date();**

**System.out.println("util date " +dateu);**

**long l =dateu.getTime();**

**java.sql.Date sqlDate = new java.sql.Date(l);**

**System.out.println("SQL date "+sqlDate);**

**}**

**Differences between java.util.Date and java.sql.Date**

**java.util.Date**

1) It is general Utility Class to handle Dates in our Java Program.

2) It represents both Data and Time.

**java.sql.Date**

1) It is specially designed Class to handle Datesw.r.t DB Operations.

2) It represents only Date but not Time.

**Note:**

=> In sql package Time class is availble to represent Time values

=> In sql package TimeStamp class is available to represent both Date and Time.

-> Inserting Date Values into Database:

Various databases follow various styles to represent Date.

**Eg:**

Oracle: dd-MMM-yy eg: 28-May-90

MySQL : yyyy-mm-dd eg: 1990-05-28

java.sql.Date => information is stored as "yyyy-mm-dd"

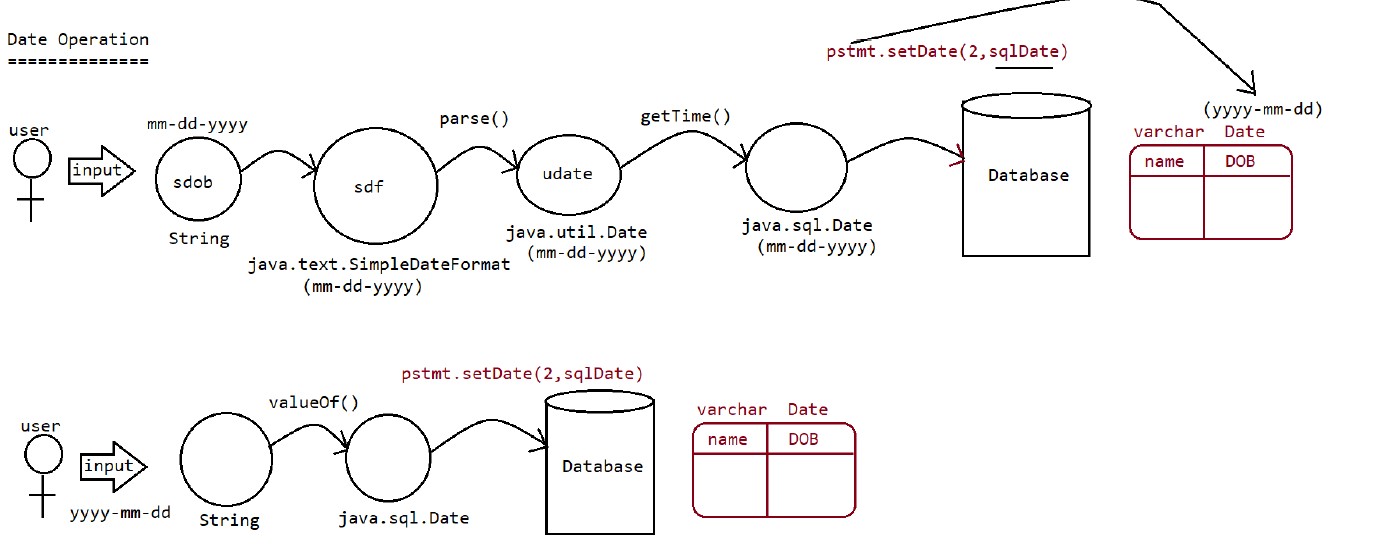
=> If we use simple Statement object to insert Date values then we should provide Date value in the database supported format,which is difficult to the programmer.

=> If we use PreparedStatement,then we are not required to worry about database supported form,

just we have to call

**pst.setDate (2, java.sql.Date);**

This method internally converts date value into the database supported format. Hence it is highly recommended to use PreparedStatement to insert Date values into database.



**Steps to insert Date value into Database:**

=> DB: create table users(name varchar2(10),dop date);

**1. Read Date from the end user(in String form)**

System.out.println("Enter DOP(dd-mm-yyyy):");

String dop=sc.next();

**2. Convert date from String form to java.util.Date form by using SimpleDateFormat object.**

SDF sdf= new SDF("dd-MM-yyyy");

java.util.Date udate=sdf.parse(dop);

**3. convert date from java.util.Date to java.sql.Date**

long l = udate.getTime();

java.sql.Date sdate=new java.sql.Date(l);

**4. set sdate to query**

pst.setDate(2,sdate);

**5. int rowAffected= pst.executeUpdate();//Execute the query.**

UserInput => SimpleDateFormat====> java.util.Date => java.sql.Date =>

ps.setDate(1,date) =>DB

|-> parse()

Program To Demonstrate Inserting Date Values Into Database:

DB: create table users(name varchar2(10),dop date);

Note:

If end user provides Date in the form of "yyyy-MM-dd" then we can convert directly

that String into java.sql.Date form as

follows...

eg:

String s = "1980-05-27";

java.sql.Date sdate=java.sql.Date.valueOf(s);

Let’s see how to insert the date into into the SQL using the code:-

**package DateOperation;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**import java.text.ParseException;**

**import java.text.SimpleDateFormat;**

**import java.util.Scanner;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class DateInsertapp {**

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

**Connection connection=null;**

**PreparedStatement pstmt=null;**

**Scanner sc = new Scanner(System.in);**

**System.out.println("Enter the name::");**

**String name =sc.next();**

**System.out.println("Enter the data Of Birth::(dd-mm-yyyy)");**

**String sDOB =sc.next();**

**SimpleDateFormat sdf=new SimpleDateFormat("dd-mm-yyyy");**

**java.util.Date uDate =sdf.parse(sDOB);**

**long time =uDate.getTime();**

**java.sql.Date sqlDate = new java.sql.Date(time);**

**System.out.println("String dob id :: "+sDOB);**

**System.out.println("Util date is :: "+uDate);**

**System.out.println("SQL data is :: "+sqlDate);**

**try {**

**connection =Connectionjdbc.getJdbcConnection();**

**String sqlInsertQuery="insert into userdata(name,dob) values(?,?)";**

**if(connection!=null) {**

**pstmt=connection.prepareStatement(sqlInsertQuery);**

**if(pstmt!=null) {**

**pstmt.setString(1, name);**

**pstmt.setDate(2, sqlDate);**

**int rowaffected=pstmt.executeUpdate();**

**System.out.println("No.of row affected "+rowaffected);**

**}**

**}**

**}catch(SQLException se) {**

**se.printStackTrace();**

**}**

**finally{**

**try {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**}**

**catch(SQLException se) {**

**se.printStackTrace();**

**}**

**}**

**}**

**}**

**To retrieve the Date from the database let’s see the code:-**

**package DateOperation;**

**import java.sql.Connection;**

**import java.sql.Date;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class SelectQuery {**

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

**Connection connection=null;**

**PreparedStatement pstmt=null;**

**ResultSet resultSet=null;**

**String name =null;**

**Date dob =null;**

**try {**

**String selectQuery="select name,dob from userdata";**

**connection=Connectionjdbc.getJdbcConnection();**

**if(connection!=null) {**

**pstmt =connection.prepareStatement(selectQuery);**

**resultSet = pstmt.executeQuery();**

**if (resultSet != null) {**

**while (resultSet.next()) {**

**name =resultSet.getString(1);**

**dob =resultSet.getDate(2);**

**SimpleDateFormat sdf = new SimpleDateFormat("dd-MM- yyyy");**

**dob = sdf.format(dob1);**

**}**

**}**

**}**

**System.out.println("Name \t"+"DOB");**

**System.out.println(name +"\t"+dob);**

**}**

**catch(SQLException se) {**

**se.printStackTrace();**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**}**

**}**

**}**

**Agenda**

🡺Blob,Clob Operation

🡺Stored Procedure(Callable Statement)

🡺Connection Pooling(Servlet,Hibernate,SpringJDBC,SpringORM,SpringJPA)

🡺Transaction

🡺javax.sql,Rowset

**BLOB=> Binary large Object(Images,videos)**

**CLOB=> Character Large Object(Text Files)**

**Working with Large Objects (BLOB And CLOB)**

Sometimes as the part of programming requirement, we have to insert and retrieve large files like

images,video files,audio files,resume etc wrt database.

Eg:upload image in matrinomial web sites upload resume in job related web sites

To store and retrieve large information we should go for Large Objects(LOBs).

There are 2 types of Large Objects.

**1. Binary Large Object (BLOB)**

**2. Character Large Object (CLOB)**

**1) Binary Large Object (BLOB)**

A BLOB is a collection of binary data stored as a single entity in the database.

BLOB type objects can be images,video files,audio files etc..

BLOB datatype can store maximum of "4GB" binary data.

eg: sachin.jpg

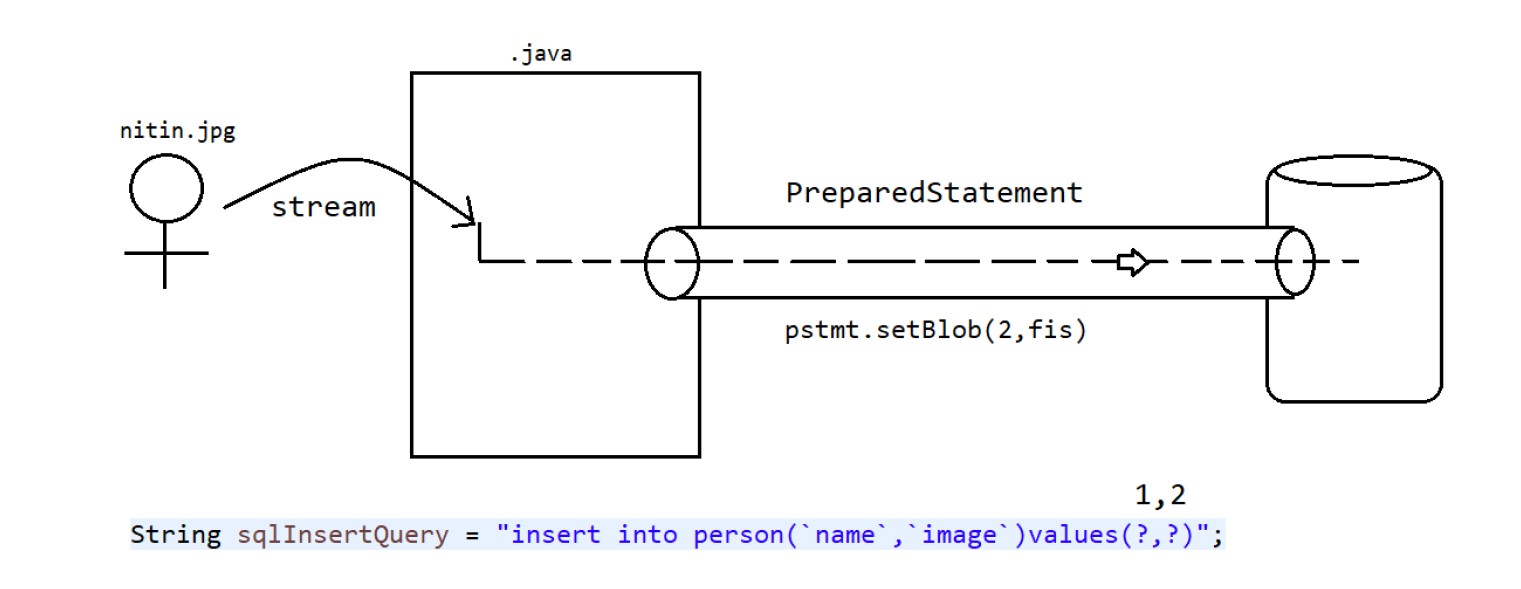
**2) CLOB (Character Large Objects):**

A CLOB is a collection of Character data stored as a single entity in the database.

CLOB can be used to store large text documents(may plain text or xml documents)

CLOB Type can store maximum of 4GB data.

eg: resume.txt



**Steps to insert BLOB type into database:**

1. create a table in the database which can accept BLOB type data.

create table persons(name varchar2(10),image BLOB);

2. Represent image file in the form of Java File object.

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

3. Create FileInputStream to read binary data represented by image file

FileInputStream fis = new FileInputStream(f)

4. Create PreparedStatement with insert query.

PreparedStatement pst = con.prepareStatement("insert into persons

values(?,?)");

5. Set values to positional parameters.

pst.setString(1,"katrina");

To set values to BLOB datatype, we can use the following method: setBinaryStream()

public void setBinaryStream(int index,InputStream is)

public void setBinaryStream(int index,InputStream is,int length)

public void setBinaryStream(int index,InputStream is,long length)

6. execute sql query

pst.executeUpdate();

**Let’s understand with the code:-**

**package in.neuron.main;**

**import java.io.File;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class BlobOperations {**

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

**Connection connection =null;**

**PreparedStatement pstmt=null;**

**try {**

**connection =Connectionjdbc.getJdbcConnection();**

**String sqlInsertQuery = "Insert into person(name,image)"**

**+ "values(?,?);";**

**if(connection!=null) {**

**pstmt=connection.prepareStatement(sqlInsertQuery);**

**pstmt.setString(1, "gautam");**

**//Image file is reaching to java application**

**File f = new File("C:\\Users\\gauta\\eclipse-workspace\\BlobClobjdbc\\src\\in\\neuron\\main\\gautam.png");**

**FileInputStream fis = new FileInputStream(f);**

**//setting the input information and sending the data to database**

**pstmt.setBlob(2, fis);**

**System.out.println("Insert image from :: "+f.getAbsolutePath());**

**int noofRows=pstmt.executeUpdate();**

**if(noofRows>=1)**

**System.out.println("Record Inserted successfully");**

**else**

**System.out.println("Record not inserted");**

**}**

**}**

**catch(SQLException se) {**

**se.printStackTrace();**

**}**

**catch(FileNotFoundException fe) {**

**fe.printStackTrace();**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**}**

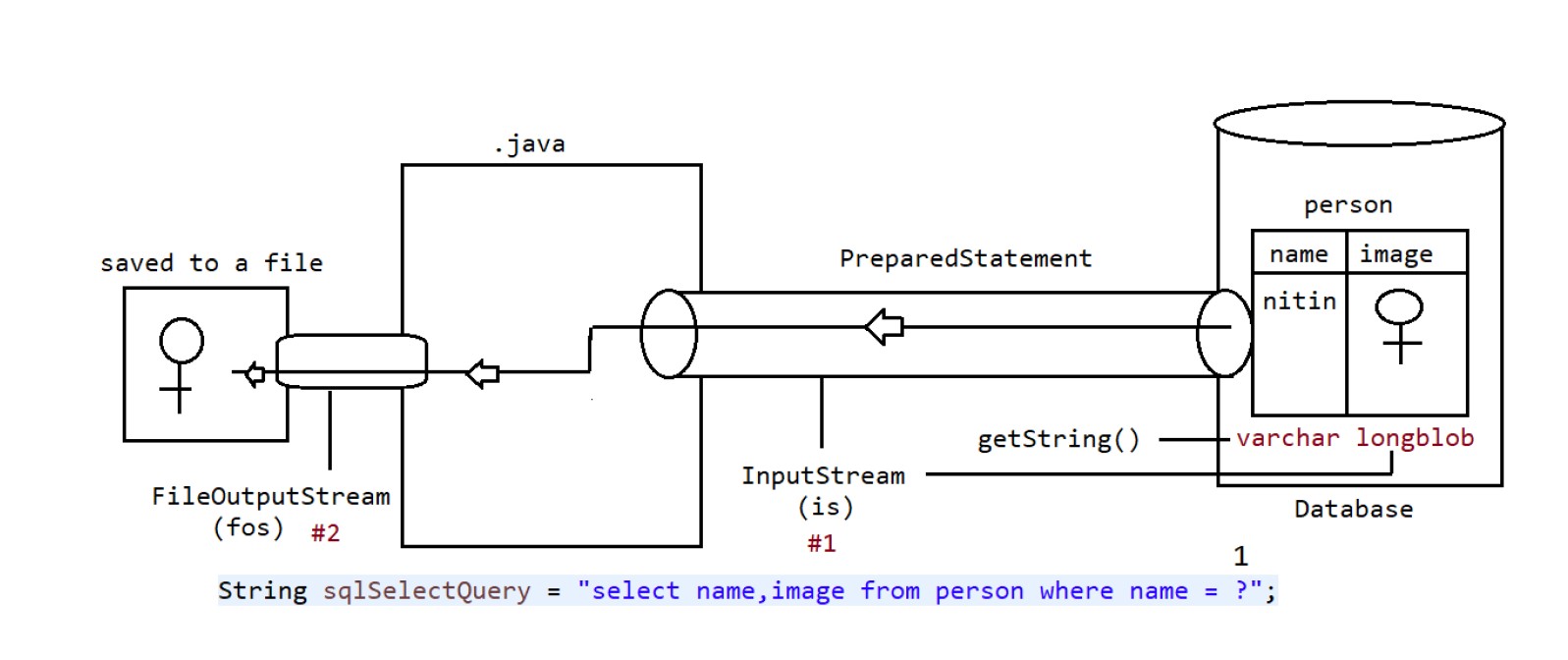
**}**

**}**

In the above code we are inserting the **Binary large object** which can only be inserted by using certain APIs.

In this case binary large object is a image file. To insert the blob in database we use FileInputStream which is one of the api which we used during deserialization of the object.

**🡺Retrieving of data from blob**



**Steps to Retrieve BLOB type from Database**

1. Prepare ResultSet object with BLOB type

ResultSet rs = st.executeQuery("select \* from persons");

2. Read Normal data from ResultSet

String name=rs.getString(1);

3. Get InputStream to read binary data from ResultSet

InputStream is = rs.getBinaryStream(2);

4. Prepare target resource to hold BLOB data by using FileOutputStream

FileOutputStream fos = new FOS("katrina\_new.jpg");

5. Read Binary Data from InputStream and write that Binary data to output Stream.

int i=is.read();

while(i!=-1)

{

fos.write(i);

is.read();

}

Or

byte[] b= new byte[2048];

while(is.read(b) > 0){

fos.write(b);

}

**Let’s see the example**

**package in.neuron.main;**

**import java.io.File;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.FileOutputStream;**

**import java.io.InputStream;**

**import java.sql.Blob;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import org.apache.commons.io.IOUtils;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class SelectBlobOpereation {**

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

**Connection connection = null;**

**PreparedStatement pstmt = null;**

**ResultSet resultSet =null;**

**String name ="gautam";**

**try {**

**connection = Connectionjdbc.getJdbcConnection();**

**String sqlInsertQuery = "select name,image from person where name =?";**

**pstmt = connection.prepareStatement(sqlInsertQuery);**

**if (connection != null && pstmt != null) {**

**pstmt.setString(1, name);**

**resultSet = pstmt.executeQuery();**

**if(resultSet.next()) {**

**String userName =resultSet.getString(1);**

**//fetching the image and keeping it in harddisk**

**InputStream is =resultSet.getBinaryStream(2);**

**String fileName ="gautam.jpg";**

**File file =new File(fileName);**

**FileOutputStream fos =new FileOutputStream(file);**

**byte [] buffer =new byte[2048];**

**while(is.read(buffer)>0) {**

**fos.write(buffer);**

**}**

**fos.flush();**

**System.out.println(userName);**

**System.out.println("File is save to the location::" +file.getAbsolutePath());**

**}**

**}**

**} catch (SQLException se) {**

**se.printStackTrace();**

**} catch (Exception e) {**

**e.printStackTrace();**

**} finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**}**

**}**

**}**

The code which is written below is IO operation code which can be replace by one of the jar

**byte [] buffer =new byte[2048];**

**while(is.read(buffer)>0) {**

**fos.write(buffer);**

**}**

The jar name is apace IO jar , and in place of this code ,after adding the code we can just write two line of code i.e.

**IOUtils.copy(is,fos);**

**CLOB (Character Large Objects)**

A CLOB is a collection of Character data stored as a single entity in the database.

CLOB can be used to store large text documents(may plain text or xml documents)

CLOB Type can store maximum of 4GB data.

Eg: resume.txt

Steps to insert CLOB type file in the database:

All steps are exactly same as BLOB, except the following differences

1. Instead of FileInputStream, we have to take FileReader.

2. Instead of setBinaryStream() method we have to use setCharacterStream() method.

public void setCharacterStream(int index,Reader r) throws SQLException

public void setCharacterStream(int index,Reader r,int length) throws SQLException

public void setCharacterStream(int index,Reader r,long length) throws SQLException

**public class clobInsert {**

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

**Connection connection = null;**

**PreparedStatement pstmt = null;**

**try {**

**connection = Connectionjdbc.getJdbcConnection();**

**String sqlInsertQuery = "Insert into cities(name,history)" + "values(?,?);";**

**if (connection != null) {**

**pstmt = connection.prepareStatement(sqlInsertQuery);**

**if (pstmt != null) {**

**pstmt.setString(1, "Bengaluru");**

**// Image file is reaching to java application**

**File f = new File(**

**"C:\\Users\\gauta\\eclipse-workspace\\BlobClobjdbc\\src\\in\\neuron\\main\\Bengaluru.txt");**

**FileReader reader = new FileReader(f);**

**// setting the input information and sending the data to database**

**pstmt.setCharacterStream(2, reader);**

**System.out.println("Insert image from :: " + f.getAbsolutePath());**

**int noofRows = pstmt.executeUpdate();**

**if (noofRows >= 1)**

**System.out.println("Record Inserted successfully");**

**else**

**System.out.println("Record not inserted");**

**}**

**}**

**} catch (SQLException se) {**

**se.printStackTrace();**

**} catch (FileNotFoundException fe) {**

**fe.printStackTrace();**

**} catch (Exception e) {**

**e.printStackTrace();**

**} finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**}**

**}**

**}**

**Retrieving CLOB Type from Database:**

All steps are exactly same as BLOB, except the following differences..

1. Instead of using FileOutputStream,we have to use FileWriter

2. Instead of using getBinaryStream() method we have to use getCharacterStream() method

**Q. What is the difference between BLOB and CLOB?**

We can use BLOB Type to represent binary information like images, video files, audio files etc

Where as we can use CLOB Type to represent Character data like text file, xml file etc...

refer: BlobApp,ClobApp

**package in.neuron.main;**

**import java.io.File;**

**import java.io.\*;**

**import java.io.FileOutputStream;**

**import java.io.InputStream;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import org.apache.commons.io.IOUtils;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class clobSelect {**

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

**Connection connection = null;**

**PreparedStatement pstmt = null;**

**ResultSet resultSet =null;**

**String name ="Bengaluru";**

**try {**

**connection = Connectionjdbc.getJdbcConnection();**

**String sqlInsertQuery = "select name,history from cities where name =?";**

**pstmt = connection.prepareStatement(sqlInsertQuery);**

**if (connection != null && pstmt != null) {**

**pstmt.setString(1, name);**

**resultSet = pstmt.executeQuery();**

**while (resultSet.next()) {**

**String userName =resultSet.getString(1);**

**//fetching the image and keeping it in harddisk**

**Reader rs =resultSet.getCharacterStream(2);**

**String fileName ="Bengaluru.txt";**

**File file =new File(fileName);**

**FileWriter writer =new FileWriter(file);**

**IOUtils.copy(rs,writer);**

**writer.flush();**

**System.out.println(userName);**

**System.out.println("File is save to the location::" +file.getAbsolutePath());**

**}**

**}**

**} catch (SQLException se) {**

**se.printStackTrace();**

**} catch (Exception e) {**

**e.printStackTrace();**

**} finally {**

**Connectionjdbc.closeConnection(null, pstmt, connection);**

**}**

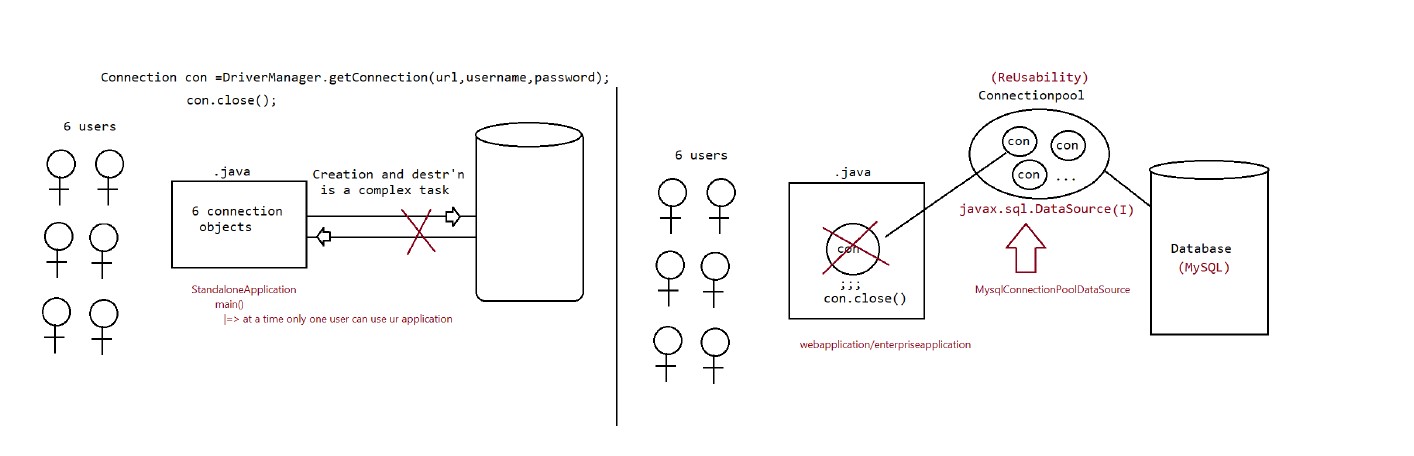
**}**

**}**

**Connection Pooling**

In Java applications, interacting with the databases using **JDBC (Java Database Connectivity)**, and managing the database **connections** efficiently is crucial for **optimal** **performance** and **resource** **utilization**. Connection pooling is the **technique** employed to achieve the goal by **reusing** the established **database** **connections** instead of creating new ones for each **request**.

JDBC usage is **establishing** the new database connection involving the costly network **communication** **process**, **authentication,** and **resource** **allocation**. This can significantly impact **application** **performance**, especially in the **interactions** of **frequent** **databases**. When an application needs to perform **database** **operations**, it will be requested to the **connection** from the pool, utilized it and it will be returned it to the pool once it is done. This approach can minimize the **overhead** of the **connection** establishment and teardown, leading to be improved **performance** and **scalability**.



🡺If we required to communicate with database multiple times then it not recommended to create separate Connection object every time, because creating and destroying Connection object every time creates performance problems.

🡺To overcome this problem, we should go for Connection Pool.

🡺Connection Pool is a poll of already created connection object which are ready to use.

🡺If we want to communicate with database then we request connection pool to provide Connection. Once we go the Connection, by using that we can communicates with database.

🡺After completing tour work, we can return Connection to the pool instead of destroying. Hence the main advantage of Connection Pool is we can reuse same Connection object multiple times, so that overall performance of application will be improved.

**Process to implement Connection Pooling:**

**1. Creation of DataSource object**

DataSource is responsible to manage connections in Connection Pool.

DataSource is an interface present in javax.sql package.

Driver Software vendor is responsible to provide implementation.

Oracle people provided implementation class name is :OracleConnectionPoolDataSource.

This class present inside oracle.jdbc.pool package and it is the part of ojdbc6.jar.

OracleConnectionPoolDataSource ds= new OracleConnectionPoolDataSource();

MySqlConnectionPoolDataSource ds= new MySqlConnectionPoolDataSource();

**2. Set required JDBC Properties to the DataSource object:**

ds.setURL("jdbc:oracle:thin:@localhost:1521:XE");

ds.setUser("scott");

ds.setPassword("tiger");

**3. Get Connection from DataSource object:**

Connection con = ds.getConnection();

Once we got Connection object then remaining process is as usual.

**Note:**

This way of implementing Connection Pool is useful for Standalone applications. In the case of web and enterprise applications, we have to use server level connection pooling. Every web and

application server can provide support for Connection Pooling.

**Q. What is the difference Between getting Connection object by using DriverManager**

**and DataSource object?**

=> In the case of DriverManager.getConnection(), always a new Connection object

will be created and returned.

=> But in the case of DataSourceObject.getConnection(), a new Connection object

won't be created and existing Connection object will be returned from Connection Pool.

**Let’s understand with the example:-**

**public class ConnectionPoolingApp {**

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

**//creating a Pooled Connection Object**

**MysqlConnectionPoolDataSource dataSource = new MysqlConnectionPoolDataSource();**

**dataSource.setUrl("jdbc:mysql://localhost:3306/ineuron");**

**dataSource.setUser("root");**

**dataSource.setPassword("Gaurav45");**

**//logical connection bought from Connection Pool**

**Connection connection=dataSource.getConnection();**

**Statement statement =connection.createStatement();**

**if(connection !=null && statement!=null) {**

**ResultSet resultSet =statement.executeQuery("select sid,sname,saddr from student");**

**System.out.println("SID\tSNAME\tsaddress");**

**while(resultSet.next()) {**

**int sid =resultSet.getInt(1);**

**String name =resultSet.getString(2);**

**String saddress =resultSet.getString(3);**

**System.out.println(sid+"\t"+name+"\t"+saddress);**

**}**

**}**

**connection.close();**

**}**

**}**

**StoredProcedure**

In our program, if we have any code which is repeatedly required, then we write that code inside function and we call that function multiple times as per our needs. So we can say that function are reusability component.

Similarly in database requirement, if we want set of SQL queries which are repeatedly used, then we write those set of statements in a single Group and we call that group based on our requirement.

This group of sql statement only we call as “Stored procedure”.

This store procedure is stored inside DB engine permanently and we need to just make a call to it.

Procedure Sytax in sql

**Create procedure ‘getStudents’(id int)**

**Begin**

**Select sid,sname,saddr**

**From student**

**Where sid =id;**

**END**

**Let’s see how to call the procedure using code in java:-**

**package in.neuron.main;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import com.mysql.cj.jdbc.CallableStatement;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class RetrieveStudentRecordApp {**

**public static void main(String[] args) {**

**Connection connection =null;**

**java.sql.CallableStatement cstmt =null;**

**ResultSet resultSet =null;**

**try {**

**connection =Connectionjdbc.getJdbcConnection();**

**String storedProcedure ="{call getStudents(?)}";**

**if(connection!=null)**

**//fetching the procedure**

**cstmt=connection.prepareCall(storedProcedure);**

**if(cstmt!=null) {**

**//before calling set the input value to StoredProcedure**

**Integer id =1;**

**cstmt.setInt(1, id);**

**//execute the stored Procedure**

**cstmt.execute();**

**resultSet =cstmt.getResultSet();**

**if(resultSet!=null) {**

**if(resultSet.next()) {**

**System.out.println(resultSet.getInt(1) +"\t"+resultSet.getString(2) +"\t" + resultSet.getString(3));**

**}**

**else {**

**System.out.println("Record not found for the id ::" +id);**

**}**

**}**

**}**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**finally {**

**try {**

**Connectionjdbc.closeConnection(resultSet, cstmt, connection);**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**}**

Let’s see another Example of Procedure:-

**CREATE DEFINER=`root`@`localhost` PROCEDURE `getStudentsById`(IN id int, out stdName varchar(20),out stdAddr varchar(20))**

**BEGIN**

**select sname,saddr into stdName,stdAddr**

**from student**

**where sid=id ;**

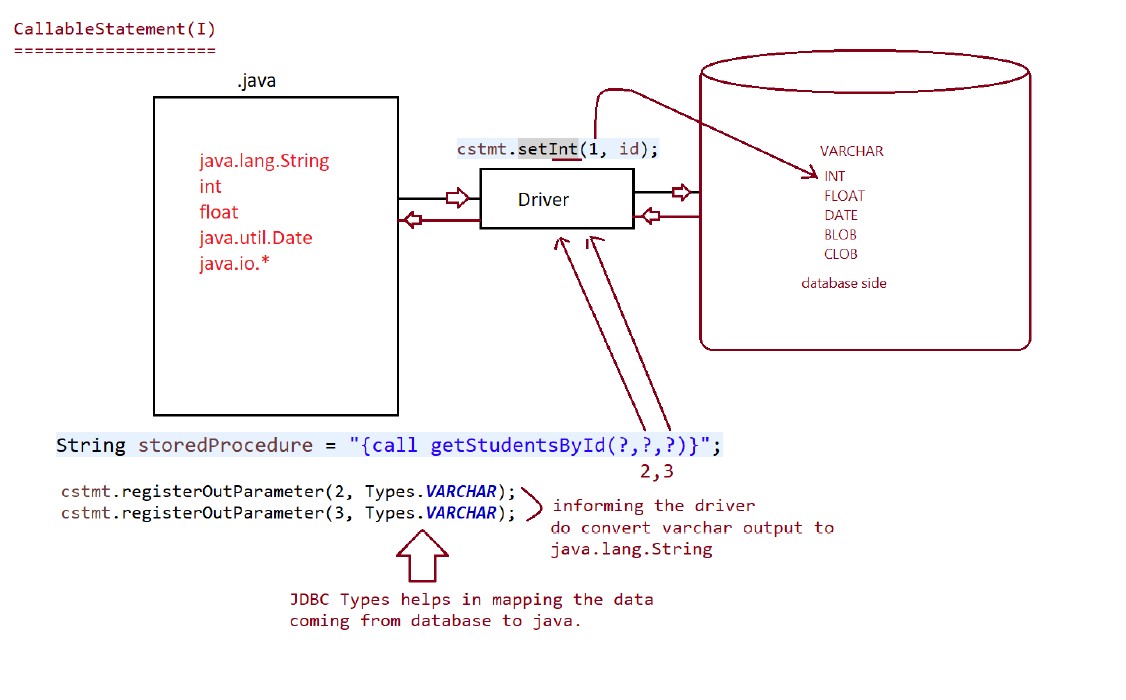
**END**

The above code is the store procedure where we red marked statements are input and output(separated with commas) of the Procedure:-

**IN** mean Input

**out** mean Input

In the procedure we are selecting sname and saddr and feeding into the stdName and stdAddr which needs to be fetched out using the JDBC Program .Let’s understand with the diagram.



As we can see in the procedure there is 1 input and 2 output but **JDBC** all input and output as input so we need to specify the output of the procedure to change into specific datatype so we need to specify in the driver that these data types are of this type we can do so as specified in the diagram.

**cstmt.registerOutParameter(2,Types.VARCHAR);**

**cstmt.registerOutParameter(3,Types.VARCHAR);**

Let’s understand with example:-

**package in.neuron.main;**

**import java.sql.Connection;**

**import java.sql.ResultSet;**

**import java.sql.Types;**

**import in.ineuron.jdbcUtil.Connectionjdbc;**

**public class RetreiveStrudentRecordusingColumnNames {**

**public static void main(String[] args) {**

**Connection connection =null;**

**java.sql.CallableStatement cstmt =null;**

**ResultSet resultSet =null;**

**try {**

**connection =Connectionjdbc.getJdbcConnection();**

**String storedProcedure ="{call getStudentsById(?,?,?)}";**

**if(connection!=null)**

**//fetching the procedure**

**cstmt=connection.prepareCall(storedProcedure);**

**if(cstmt!=null) {**

**//before calling set the input value to StoredProcedure**

**Integer id =1;**

**cstmt.setInt(1, id);**

**cstmt.registerOutParameter(2, Types.VARCHAR);**

**cstmt.registerOutParameter(3, Types.VARCHAR);**

**cstmt.execute();**

**System.out.println("Name of the Student is ::"+cstmt.getString(2));**

**System.out.println("address of the Student is ::"+cstmt.getString(3));**

**}**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**finally {**

**try {**

**Connectionjdbc.closeConnection(resultSet, cstmt, connection);**

**}**

**catch(Exception e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**}**

**Transaction**

🡺Process of combining all related operations into a single unit and executing on the rule "either all or none", is called transaction management.

🡺Hence transaction is a single unit of work and it will work on the rule "either all or none".

Case-1: Funds Transfer

1. debit funds from sender's account

2. credit funds into receiver's account

All operations should be performed as a single unit only. If debit from sender's account completed and credit into receiver's account fails then there may be a chance of data inconsistency problems.

Case-2: Movie Ticket Reservation

1. Verify the status

2. Reserve the tickets

3. Payment

4. issue tickets.

All operations should be performed as a single unit only. If some operations success and some operations fails then there may be data inconsistency problems.

**Transaction Properties:**

Every Transaction should follow the following four ACID properties.

**1. A ➔ Atomiticity**

Either all operations should be done or None.

**2. C ➔ Consistency(Reliabile Data)**

It ensures bringing database from one consistent state to another consistent state.

**3. I ➔ isolation (Seperatation)**

Ensures that transaction is isolated from other transactions

**4. D ➔ Durability**

It means once transaction committed, then the results are permanent even in the case of system restarts, errors etc.

**Types of Transactions**

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

There are two types of Transactions

**1. Local Transactions**

**2. Global Transactions**

**1. Local Transactions:**

All operations in a transaction are executed over same database. Eg: Funds transfer from one account to another account where both accounts in the same bank.

**2. Global Transactions:**

All operations is a transaction are expected over different databases. Eg: Funds Transfer from one account to another account and accounts are related to different banks.

**Note:**

JDBC can provide support only for local transactions. If we want global transactions then we have to go for EJB(Enterprise Java Bean) or Spring framework.

Let’s see what is the usage of .properties file in the jdbc.

.properties is generally used to write sql query or the database value that might change during execution of the jdbc utility file so let’s understand with the example:-

Let’s see first the utility file code here:-

**package in.neuron.jdbcutil;**

**import java.io.FileInputStream;**

**import java.io.IOException;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Statement;**

**import java.util.Properties;**

**public class Conectionjdbc {**

**// Private constructor to prevent instantiation**

**private Conectionjdbc() {}**

**public static Connection getJdbcConnection() throws SQLException, IOException {**

**Connection connection = null;**

**Properties props = new Properties();**

**// Load properties from a file**

**try (FileInputStream fis = new FileInputStream("src/in/neuron/properties/jdbc.properties")) {**

**props.load(fis);**

**} catch (IOException e) {**

**throw new IOException("Error loading JDBC properties file.", e);**

**}**

**// Retrieve connection properties from the loaded properties**

**String url = props.getProperty("url");**

**String id = props.getProperty("user");**

**String pass = props.getProperty("password");**

**// Establish the connection**

**connection = DriverManager.getConnection(url, id, pass);**

**if (connection != null) {**

**return connection;**

**} else {**

**throw new SQLException("Failed to establish a JDBC connection.");**

**}**

**}**

**public static void closeConnection(ResultSet rs, Statement st, Connection con) throws SQLException {**

**if (st != null) {**

**st.close();**

**}**

**if (con != null) {**

**con.close();**

**}**

**if (rs != null) {**

**rs.close();**

**}**

**}**

**}**

In the utility file, we are calling all the properties file use file stream and after the call of the properties file using file stream we can see that it is fetching the value of username ,id, password so let’s see what is written in the properties file of the java:

**url =jdbc:mysql://localhost:3306/ineuron**

**user =root**

**password =Gaurav45**

Properties file are simple file like .txt in which type all the value of sql and database credentials

**Now Let’s see how to do transaction in java using JDBC. Before understanding the transaction in the JDBC we first need to understand that connection are auto commit statement and if the transaction fails in between that it will auto commit the partial transaction which do not follow the properties of ACID, so first we need to set the autoCommit false in the connection.**

**Process of Transaction Management in JDBC:**

1. Disable auto commit mode of JDBC

By default auto commit mode is enabled. i.e. after executing every SQL Query, the changes will be committed automatically in the database.

We can disable auto commit mode as follows

Con.SetAutoCommit(false);

2. If all operations completed then we commit the transaction by using the following method con.commit()

3. If any sql fails then we have to rollback operations which are already completed by using rollback() method. con.rollback().

Program to demonstrate Transaction app:-

**package TransactionApp;**

**import com.mysql.cj.protocol.Resultset;**

**import in.neuron.jdbcutil.Conectionjdbc;**

**public class TransactionApp {**

**public static void main(String[] args) {**

**Connection connection =null;**

**PreparedStatement pstmt =null;**

**Statement stmt =null;**

**ResultSet rs =null;**

**ResultSet rs1 =null;**

**try {**

**connection =Conectionjdbc.getJdbcConnection();**

**stmt = connection.createStatement();**

**System.out.println("Data before transaction");**

**rs =stmt.executeQuery("select name,balance from accounts");**

**while (rs.next()) {**

**System.out.println(rs.getString(1) +"\t"+ rs.getInt(2));**

**}**

**connection.setAutoCommit(false);**

**System.out.println();**

**System.out.println("DATA AFTER TRANSACTION");**

**stmt.executeUpdate("update accounts set balance=balance-3000 where name ='sachin';");**

**stmt.executeUpdate("update accounts set balance=balance+3000 where name ='sachin';");**

**System.out.println("Do you really want to commit yes/no");**

**Scanner sc = new Scanner(System.in);**

**String option =sc.next();**

**if(option.equalsIgnoreCase("yes")){**

**connection.commit();**

**}**

**else**

**{**

**connection.rollback();**

**System.out.println("transaction roll back");**

**}**

**rs1 =stmt.executeQuery("select name,balance from accounts");**

**while (rs1.next()) {**

**System.out.println(rs1.getString(1) +"\t"+ rs1.getInt(2));**

**}**

**}**

**catch(SQLException e) {**

**e.printStackTrace();**

**}**

**catch(IOException j) {**

**j.printStackTrace();**

**}**

**catch(Exception k) {**

**k.printStackTrace();**

**}**

**finally {**

**try {**

**Conectionjdbc.closeConnection(null, pstmt, connection);**

**}**

**catch(SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**}**

**Savepoint(I)**

🡺Savepoint is an interface present in java.sql.package.

🡺Introduced in JDBC 3.0 version.

🡺Driver Software Vendor is responsible to provide implementation.

🡺Savepoint concept is applicable only in Transactions.

🡺Within a Transaction if want to rollback a particular group of operation based on some condition then we should go for savepoint.

🡺We can set Savepoint by using setSavepint() method of connection interface. Savepoint sp = con.setSavepoint();

🡺To perform rollback operation for a particular group of operation wrt Savepoint, we can use rollback() method as follows con.rollback(sp);

🡺We can release or delete Savepoint by using release Savepoint() method of Connection interface. Con.releaseSavepoint(sp);

Let’s see with the examples:-

con.setAutoCommit(false)

operation-1

operation-2

operation-3

SavePoint sp =new SavePoint();

Operation-4

Operation-5

If(balance<=1000)

Con.rollback(sp)

Else

Con.releaseSavePoint();

Operation-6

Con.commit();

At line-1 if balance<10000 then operation 4 and 5 will be Rollback, otherwise all operation will be performed normally.

**package TransactionApp;**

**import java.io.FileInputStream;**

**import java.io.IOException;**

**import java.sql.Connection;**

**import java.sql.PreparedStatement;**

**import java.sql.ResultSet;**

**import java.sql.SQLException;**

**import java.sql.Savepoint;**

**import java.sql.Statement;**

**import java.util.Properties;**

**import java.util.Scanner;**

**import com.mysql.cj.protocol.Resultset;**

**import in.neuron.jdbcutil.Conectionjdbc;**

**public class SavePointApp {**

**public static void main(String[] args) {**

**Connection connection =null;**

**Statement stmt =null;**

**ResultSet rs =null;**

**try {**

**//Getting the database the connection using utilty code**

**connection =Conectionjdbc.getJdbcConnection();**

**stmt =connection.createStatement();**

**connection.setAutoCommit(false);**

**stmt.executeUpdate("insert into politicians(name,party) values('modi','BJP')");**

**stmt.executeUpdate("insert into politicians(name,party) values('stalin','dmk')");**

**stmt.executeUpdate("insert into politicians(name,party) values('babu','tdp')");**

**Savepoint sp =connection.setSavepoint();**

**stmt.executeUpdate("insert into politicians(name,party) values('rahul','BJP')");**

**System.out.println("'Opps',Something went wrong need to rollback");**

**connection.rollback(sp);**

**connection.commit();**

**rs =stmt.executeQuery("select name,party from politicians");**

**while (rs.next()) {**

**System.out.println(rs.getString(1)+"\t"+rs.getString(2));**

**}**

**}**

**catch(SQLException e) {**

**e.printStackTrace();**

**}**

**catch(IOException j) {**

**j.printStackTrace();**

**}**

**catch(Exception k) {**

**k.printStackTrace();**

**}**

**finally {**

**try {**

**Conectionjdbc.closeConnection(null,stmt, connection);**

**}**

**catch(SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**}**

Note:

Some drivers won't provide support for Savepoint. Type-1 Driver won't provide support, but

Type#4 Driver can provide support.

Type-4 Driver of Oracle provide support only for setSavepoint() and rollback() methods but not for releaseSavepoint() method.

Transaction Concurrency Problems: Whenever multiple transactions are executing concurrently then there may be a chance of transaction concurrency problems.

The following are the most commonly occurred concurrency problems.

1. Dirty Read Problem

2. Non Repeatable Read Problem

3. Phantom Read Problem

**1. Dirty Read Problem:**

Also known as uncommitted dependency problem. Before committing the transaction, if its intermediate results used by any other transaction then there may be a chance of Data inconsistency problems. This is called Dirty Read Problem.

nitin:50000

T1:update accounts set balance=balance+50000 where name='nitin'

T2:select balance from accounts where name='nitin'

T1: con.rollback();

At the end, T1 point of view, nitin has 50000 balance and T2 point of view nitin has 1Lakh. There may be a chance of data inconsistency problem. This is called Dirty Read Problem.

**2. Non-Repeatable Read Problem:**

For the same Read Operation, in the same transaction if we get different results at different times,then such type of problem is called Non-Repeatable Read Problem.

Eg:

T1: select \* from employees;

T2: update employees set esal=10000 where ename='nitin';

T1: select \* from employees;

In the above example Transaction-1 got different results at different times for the same query.

**3. Phantom Read Problem:**

A phantom read occurs when one transaction reads all the rows that satisfy a where condition and second transaction insert a new row that satisfy same where condition. If the first transaction reads for the same condition in the result an additional row will come. This row is called phantom row and this problem is called phantom read problem.

T1: select \* from employees where esal >5000;

T2: insert into employees values(300,'ravi',8000,'hyd');

T1: select \* from employees where esal >5000;

In the above code whenever transaction-1 performing read operation second time, a new row will come in the result.

To overcome these problems we should go for Transaction isolation levels.

Connection interface defines the following 4 transaction isolation levels.

1. TRANSACTION\_READ\_UNCOMMITTED ➔ 1

2. TRANSACTION\_READ\_COMMITTED ➔ 2

3. TRANSACTION\_REPEATABLE\_READ ➔ 4

4. TRANSACTION\_SERIALIZABLE ➔ 8

1. TRANSACTION\_READ\_UNCOMMITTED:

It is the lowest level of isolation. Before committing the transaction its intermediate results can be used by other transactions. Internally it won't use any locks. It does not prevent Dirty Read Problem, Non Repeatable Read Problem and Phantom

Read Problem.

We can use this isolation level just to indicate database supports transactions. This isolation level is not recommended to use.

**2. TRANSACTION\_READ\_COMMITTED:**

This isolation level ensures that only committed data can be read by other transactions.

It prevents Dirty Read Problem. But there may be a chance of Non Repeatable Read Problem and Phantom Read Problem.

**3. TRANSACTION\_REPEATABLE\_READ:**

This is the default value for most of the databases. Internally the result of SQL Query will be locked for only one transaction. If we perform multiple read operations, then there is a guarantee that for same result. It prevents Dirty Read Problem and Non Repeatable Read Problems. But still there may be a chance of Phantom Read Problem.

**4. TRANSACTION\_SERIALIZABLE:**

It is the highest level of isolation.

The total table will be locked for one transaction at a time. It prevents Dirty Read, Non-Repeatable Read and Phantom Read Problems. Not Recommended to use because it may creates performance problems.

**Note:**

Connection interface defines the following method to know isolation level. getTransactionIsolation()

Connection interface defines the following method to set our own isolation level. setTransactionIsolation(int level)

Eg:

System.out.println(con.getTransactionIsolation());

con.setTransactionIsolation(8);

System.out.println(con.getTransactionIsolation());

**Note:**

For Oracle database, the default isolation level is: 2(TRANSACTION\_READ\_COMMITED). Oracle database provides support only for isolation levels 2 and 8. For MySql database, the default isolation level is: 4(TRANSACTION\_REPEATABLE\_READ). MySql database can provide support for all isolation levels (1, 2, 4 and 8).

**Note:**

ResultSet(holds the data which is used for reading purpose)

|=> Using resultset we have just performed read operation(best suited)

|=> Is it possible to perform update,inserte and delete operation(possible but not recommended)

**(J2EE ,JEE, J2SE)(Java Enterprise Edition)**

Inside this there is two this

🡺Servlet

🡺JSP

**Servlet**

Java Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, and then send a response back to the web server.

We can use Servlet in two ways:-

* Directly
* InDirectly(By using Spring and Spring boot)

There are two two types of application:-

🡺**Stand Alone Application :-** Those application which do not require internet to run is called standalone application. It also means that if any application which is not performing client server architecture is called Stand alone Application.

**🡺Enterprise Application:-** Those application which require internet to run is called Enterprise application. It follows the client server architecture which mean application is stored in remote server and serving it’s application using DNS and IP address. It is divided into two parts:-

* **Web App**
* **Distributed App**

**Web App:-** A web App is a software application accessed through a web browser, typically running on a web server. All the business logic , controller and handler are written in same web server. Here only browser is a client.

**Distributed App:-**  A distributed app is a software application where different components run on multiple computer across a network , working together to achieve a share goal. But here browser as well as server is also a client. Suppose if we are purchasing something in flipkart and we are paying through phonepe so we are only interacting with flipkart, flipkart is interacting with phonepe to achieve the transaction.

To develop web app we are going to use servlet in which request and response is handled by the thread and as we know that thread is light weight sub process so whenever client request any thing in web application , servlet assign a light weight thread which look after the need of the client.

**Response**

* **Static Response**
* **Dynamic Response**

**Static Response:-** A static response refer to a fixed , unchanging response to client from the server for ex:- if two people are try ing to open the flipkart link in the web browser then the server will show the same webpages to the people without changing the html page.

**Dynamic Response:-**  A dynamic response refers to the response which vary and adapt with the request or time after some computation. Like suppose if two people are searching for the fare of aeroplane for some destination and if one person search 10 sec before the other then the fare for the both person for same flight will be different as the fare of the flight is volatile in nature or we can say that server is computing after every request from the client.

**Client Server Architecture**

Client-server architecture is a computing model that separates tasks between clients and servers. Clients request services from servers, which then process the requests and send back the results.

How it works

1. A client, like a web browser or mobile app, requests data or services from a server
2. The server processes the request and sends back the data or services.
3. The client renders the data or services, such as displaying a website

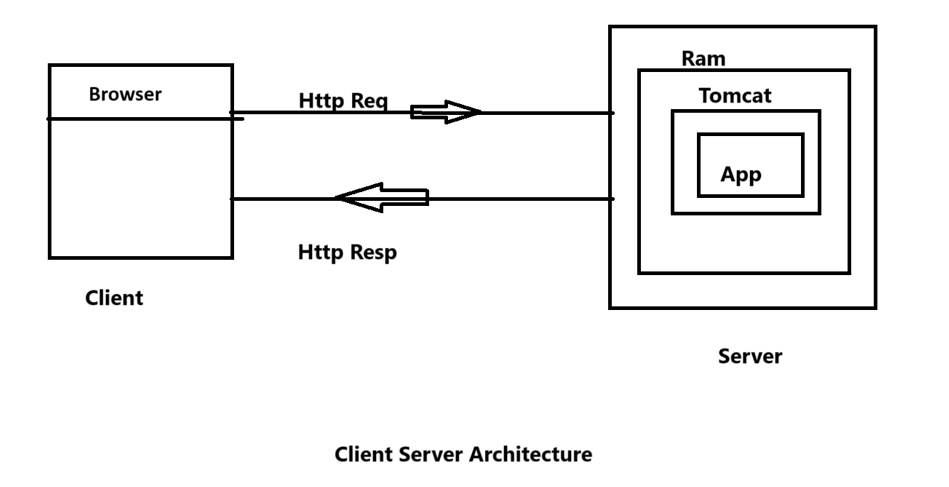
**Benefits**

* **Centralized data**: Servers can store and manage data, which can improve security
* **Scalability**: Developers can scale infrastructure up or down with minimal downtime
* **Application deployment**: Developers can deploy applications close to end users

**Requirement for Client-Server Architecture**

1. Client Machine
2. Server Machine
3. Protocol🡺 sets of Rule and regulation to be follow like TCP/IP,UDP,HTTP

In our architecture we are going to use HTTP because it is more secure, logical connection and restless in nature. It only works on current request and don’t bother to response on older request and if we need to response on older request we generally used session tracking.



In order to establish connection using the protocol **URI** Comes into picture

URI (Uniform Resource Identifier)

|

|

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

| |

| |

URN(Uniform Resource Name) URL(Uniform Resource Locator)

We generally user used URL to establish connection using Protocols. In URL, we Generally pass string that contains

**Protocol||IP address:portnumber//projectName/file name** like http//:192.12.0.49:4080/nameApplication/pageName.

In the real world scenario , ip address,port number and project name gets hidden using DNS(Domain Name System) which is easy to memorize by the user, so generally server side company purchase DNS from different host company to hide ip address and port number.

**Server**

Server are special software which are installed in the machine for serving an application or deploying an application over the web or internet. Like Tomcat,wildfy,jboss.

Work of server

🡺Accept the Request, Process the Request and identify the Req Resource and process the Resource

🡺Response back to the client

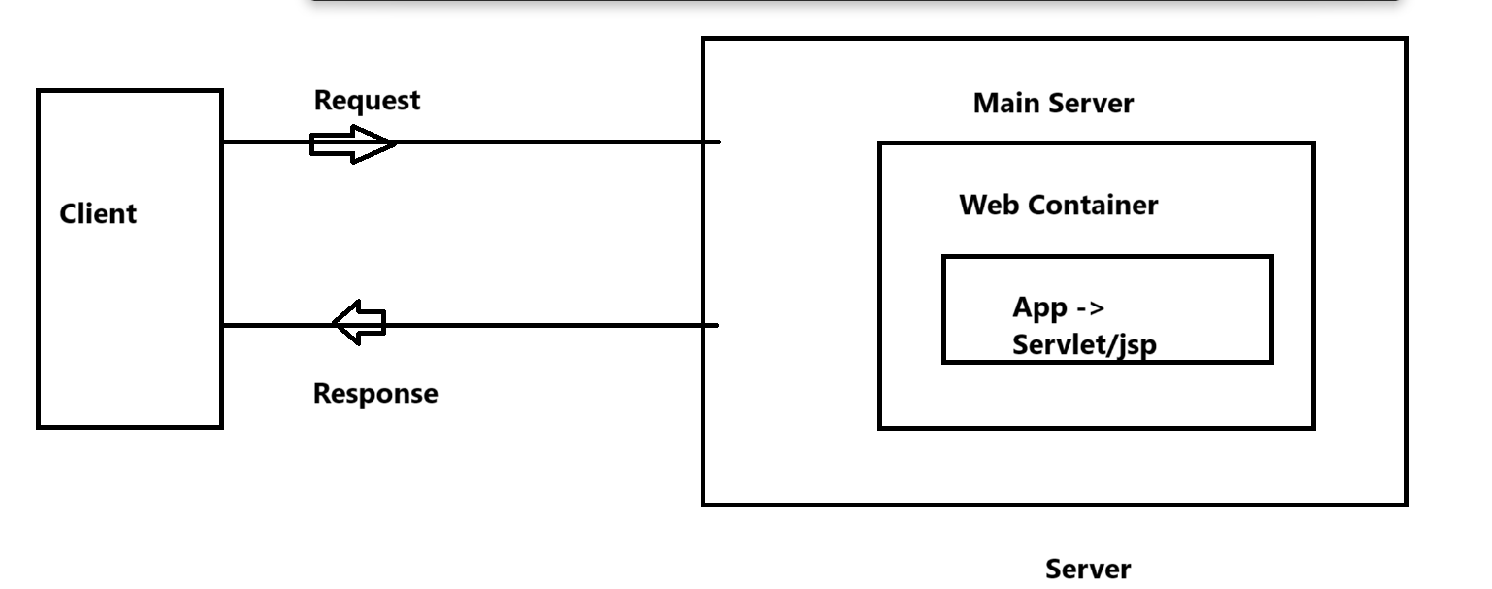
Whenever we install any of the Server it comes with two part

* **Main server**
* **Web Container**

As we can see in the diagram, In server there is two part that is main server and web container, the purpose of the main server to check the request of the client , if the request of the client is proper then only it will allow to enter in the web container,. Inside web container app is installed so according to the request of the client web container process the request and gives the response.

Generally, web container is used for dynamic response where there is need to process the request of the client. For static response there is no need of the web container as there is not any process for the response of the request.

Let’s see first how java program works:-



Java prg🡺javac🡺.class file🡺jvm🡺run

For java program to run, we need main method as jvm don’t understand from where it need to start but it case of server jvm is running the program web container is executing the code which is written in Servel or JSP so we generally don’t need main method to run the application.

**Servlet and JSP difference**

**Servlet**

* It is build on the top of java
* It takes the request ,process the request and respond back the request

**JSP**

* It is not dependent on java, it is based on html, xml, tag based
* Using JSP we can do every thing that servlet perform but only recommended to perform the response of the request.

**To Develop an application we follow Design pattern i.e MVC(Model View Controller)**

**M –Model🡺** we need to write the class or such program to interact with databases

**V-View🡺** All the code required for display(HTML,CSS,JS)

**C-Controller🡺** Accept the request and process it and act as a medium between view and model.

**Model and view** don’t interact with each ,to connect we need controller.

Let’s see how to start a servlet application, first we need to change the perspective of eclipse from java to JavaEE perspective

Now We need to create dynamic web application, and check if Tomcat server is available or not. Then we one project will be created inside it there will lots of like java Resource and source

Inside java Source we will add servlet class and in the source tab it has web-INF where we will create JSP page and html page so first all let’s see how we will create servlet page.

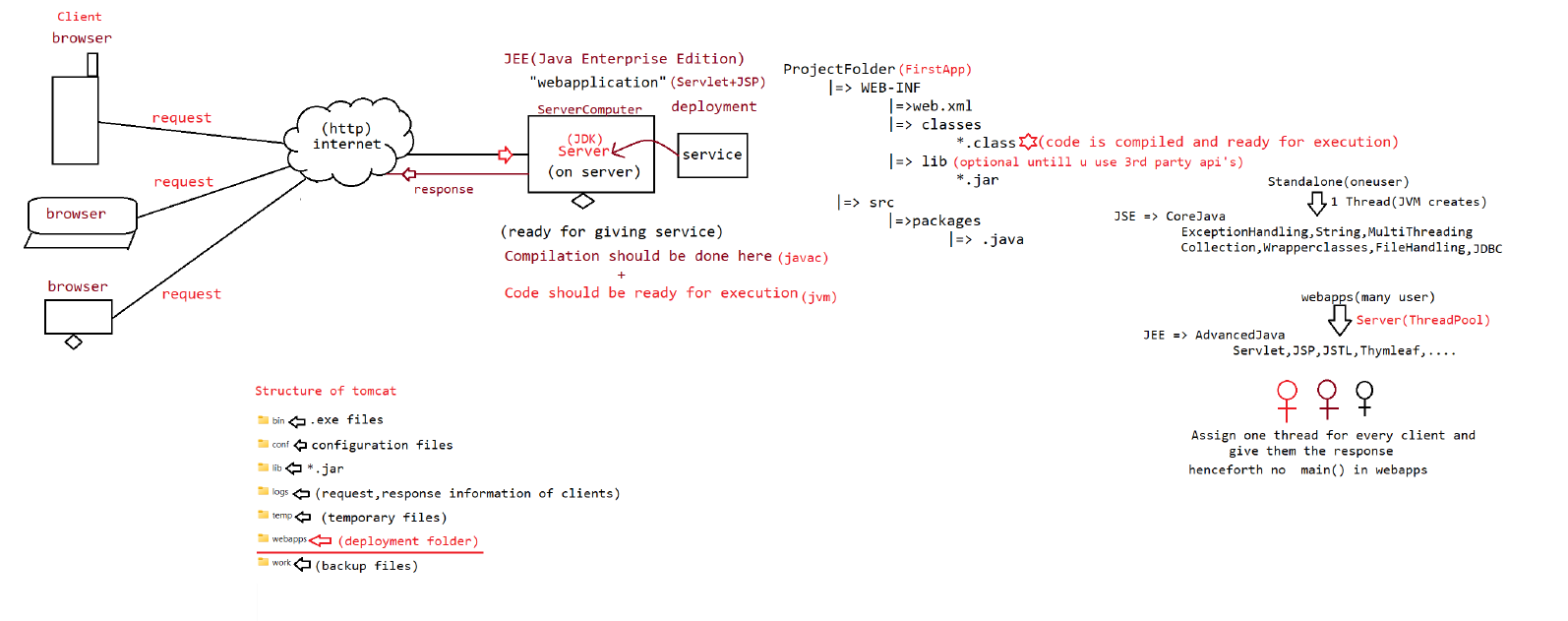
**Creation of Servlet:-**

* **By extending HttpServlet**

Servlet has three lifecycle or three methods:-

* **public void init()**
* **public void Service(HttpServletRequest req HttpServletResponse resp)**
* **public void destroy()**

Actually whatever the task is in servlet, it should be written inside the service method it takes two parameter HttpServletRequest req and response parameter i.e HttpServletResponse Resp

****

Now, suppose There is any java program which only accept the request and gives the response. Will that java program will be normal java program. The answer is no.

For this ,JEE Community had given the rule in the form of an api called servlet. This api is given by sun Microsystem. Now what is servlet api.

Servlet api is nothing but a set of interface and helper class, now again the question arises if it is a set of interface and helper classes then who is going to give the implementation of servlet api. Simple the answer is server which is connected with the servlet like Tomcat, wildfy, glassfish, JBoss.

Now if the implementation of servlet is being given by the server then where we need to keep the code of servlet , the answer is simple inside server whether it is a tomcat or jboss or wildfy.

Now question arises, how we are going to connect the application which takes the request and gives back the response, there is three ways to create the application:-(Interview Question)

1. **By using Servlet Interface (javax.servlet)**
2. **By using GenricServlet (Abstract class) (javax.servlet)**
3. **By using HttpServlet(Abstract class) (javax.servlet.http)**

So first we are going to use servlet interface to create an application, let’s see how

So First of all we are going to create the folder inside the Tomcat webapps folder with the name FirstApp inside FirstApp there will be two folders namely

* WEB-INF
* classes

Directly inside the FirstApp we will create the application with the name FirstServlet

Inside the Servlet, there is lifecycle methods namely

* Public void intit(ServletConfig Config)
* Public void service(ServletRequest request ,ServletResponse response)
* Public void destroy()
* Public servletConfig getServletConfig()
* Public String getServletInfo()

Let’s see how we are going to create the program:-

import javax.servlet.\*;

import java.io.\*;

public class FirstSevlet implements Servlet

{

static{

System.out.println("Servlet Loading");

}

public FirstServlet(){

System.out.println("Servlet Initantiation");

}

//life cycle method

public void init(ServletConfig config) throws ServletException{

System.out.println("Servlet is initialized ");

}

public void service (ServletRequest request ,ServletResponse response)throws ServletException{{

System.out.println("Providing the service for the user ");

}

public void destroy(){

System.out.println("Service DeInstantiation");

}

public ServletConfig getServletConfig(){

return null;

}

public String getServletInfo(){

return "Developed by Gaurav";

}

}

Now to execute the program we need a web.xml which needs to saved WEB-INF folder of the app and the structure of the web.xml.

<web-app>

<servlet>

<servlet-name>DemoServlet</servlet-name>

<servlet-class>FirstSevlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>DemoServlet</servlet-name>

<url-pattern>/test</url-pattern>

</servlet-mapping>

</web-app>

Now ,let’s see how the control goes when we try to execute the FirstServlet application, before knowing the control we first need to understand the function of server. Server is divided into two parts i.e.

* Main server:- accept the request
* Web container:-process the request and give back the response.

To give the request to the server , we generally use http protocol and the pattern of the http protocol is like :-

**http||IP address:portnumber//projectName/file name**  so in our case http request will be :-

<http://localhost:8080/FirstApp/test>

so the flow of control will first go the the Main server which checks if all the details are correct or not and pass the control to the container to process the request.

when we provide the request it will go directly to the web.xml of FirstApp which is present inside the WEB-INF .Inside web.xml control will first go to servletMapping block and try to find <url-pattern> test</url-pattern>, after finding the url pattern , in the same servlet mapping block there is servlet name present . .i.e. DemoServlet.

After finding the DemoServlet name, the control will go to servlet block where servlet name is present. And with servlet name ,ServletClass is also there , so the control will take the name of servlet class and execute that class.

So the flow of control is like this

(<http://localhost:8080/FirstApp/test>) 🡺 <servlet-mapping> 🡺 <url-pattern> 🡺 <servlet-name>🡺 <servlet> 🡺 <servlet-name> 🡺<servlet-class> then class will be loaded by the web container.

Q. How to load the .class file dynamically based on the runtime Requirement?

Ans:- Class c – Class.forName(“FirstServlet”);

After loading the class. Class will be instantiated by the container, and zeroArgument constructor is instantiated.

Q How to instantiated the object for the dynamic loaded class?

FirstServlet obj =(FirstServlet) c.newInstance();🡺 object is created with zero argument constructor

After Instantiation Life Cycle methods would be excecuted:-

First init method is executed like this obj.init(ServletConfig config), Now through dependency injection ServletConfig object is injected in the init method. After that body will be executed.

Now service method i.e. service(SR request,SR Response) will be executed by obj and their parameter are injected through dependency injecton and body will be executed and atlast object will call the destroy method which will deinstantiate the object.



EveryThing is shown in the diagram. In tomcat , the name of the container is Catalina.

Whenever we provide the request for the first time, loading time is bit longer than the second time to reduce the loading time we will add one more tag in the web.xml i.e. <load-on-startup>10</load-on-startup> in the servlet tag where sevlet-class is defined.

<servlet>

<servlet-name>DemoServlet</servlet-name>

<servlet-class>FirstSevlet</servlet-class>

<load-on-startup>10</load-on-startup>

</servlet>

It will decrease the time of the loading in the server of the particular servlet.

import javax.servlet.\*;

import java.io.\*;

public class FirstServlet implements Servlet

{

static{

System.out.println("Servlet Loading");

}

public FirstServlet(){

System.out.println("Servlet Initantiation");

}

//life cycle method

public void init(ServletConfig config) throws ServletException{

System.out.println("Servlet is initialized ");

}

public void service (ServletRequest request ,ServletResponse response)throws ServletException{{

//request processing logic

System.out.println("Providing the service for the user ");

//1.set the response type to html

response.setContentType("text/html");

//2. use PrintWriter object to send the response

PrintWriter out = response.getWriter();

out.println("<html><head><title> Output</title></head>");

out.println("<body>");

out.prinln("<marquee>Welcome to servlet coding....</marquee>");

out.println("</body>");

out.println("</html>");

//3.close the printer

out.close();

}

public void destroy(){

System.out.println("Service DeInstantiation");

}

public ServletConfig getServletConfig(){

return null;

}

public String getServletInfo(){

return "Developed by Gaurav";

}

}

Servlet application with Annotation:- To make the servlet using annotation, we only need to add this annotation

**@WebServlet(urlPatterns={“/test”},loadOnStartup =1)**

Above the java application like this

**@WebServlet(urlPattern={“/test”}**

import javax.servlet.\*;

import java.io.\*;

@WebServlet(urlPattern ={"/test"},loadOnStartUp=1)

public class SecondServlet implements Servlet

{

static{

System.out.println("Servlet Loading....");

}

public SecondServlet(){

System.out.println("Servlet Instantiation...");

}

//life cycle methods

public void init(ServletConfig config)throws ServletException{

//Servlet Initialization phase

System.out.println("Servlet is initialized...");

}

public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException{

//Request-Processing-logic

System.out.println("providing the service for the users");

// 1. Set the response type to HTML

response.setContentType("text/html");

// 2. Use PrintWriter object to send the response

PrintWriter out = response.getWriter();

out.println("<html><head><title>Output</title></head>");

out.println("<body>");

out.println("<h1 style='color:red'><marquee>Welcome to servlet coding...</marquee></h1>");

out.println("</body>");

out.println("</html>");

// 3. Close the PrintWriter

out.close();

}

public void destroy(){

// Deinstantiation activities

System.out.println("Servlet DeInstantiation....");

}

public ServletConfig getServletConfig(){

return null;

}

public String getServletInfo(){

return "Developed by nitin";

}

}

We don’t need to add any web.xml file to configure the servlet simple using this annotation we can map so whenever there is http request it simply

(<http://localhost:8080/FirstApp/test>)

lifecycle methods namely

* Public void intit(ServletConfig Config) :- It is used for initialisation
* Public void service(ServletRequest request ,ServletResponse response) :-process the request and gives the output
* Public void destroy():-Destroy or deinstantiation.

It simply check the annotation in java and load the class in the container to execute the code.

1. Creation of Servlet **By using GenricServlet (Abstract class)**

Now the Question Arises why we come with **GenricServlet**, as we already have servlet Interface in our bucket. The answer is simple in Servlet interface there are five methods namely

1. Intit(SC)
2. Service(SReq,SRes)
3. Destroy()
4. getServletConfig()
5. getServletInfo()

Out of which only first three are life cycle method , which we need to initialise but as servlet is an interface we need to initialise all so what sun microsystem has done it that it created an abstract class i.e. GenricServlet which implements Servlet Interface. It simple define the four methods and leaves the only service method as abstract so we need to define only service method which is responsible for processing the request and giving the response.

**Servlet(I) (5 methods)**

**|**

**|**

**GenricServlet(AC) (4 Methods Define)**

**(1 Method abstract)**

**|**

**|**

**MyServle(Concrete Class) (Service (SReq, SResp)**

This GenricServlet is implementing two interface:-

1. Serializable(I)
2. ServletConfig(I)

Let’s see the code:-

**import javax.servlet.\*;**

**import java.io.\*;**

**@WebServlet(urlPattern ={"/test"})**

**public class ThirdServlet extends GenricServlet**

**{**

**static{**

**System.out.println("Servlet Loading....");**

**}**

**public ThirdServlet(){**

**System.out.println("Servlet Instantiation...");**

**}**

**public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException{**

**//Request-Processing-logic**

**System.out.println("providing the service for the users");**

**// 1. Set the response type to HTML**

**response.setContentType("text/html");**

**// 2. Use PrintWriter object to send the response**

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

**out.println("<html><head><title>Output</title></head>");**

**out.println("<body>");**

**out.println("<h1>Genric Servlet is loading</h1>");**

**out.println("</body>");**

**out.println("</html>");**

**// 3. Close the PrintWriter**

**out.close();**

**}**

**}**

When we try to deploy the app using eclipse , eclipse will store the files in my workspace in this location

**E:\myWorkspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps**

And this type of deployment is called soft-deployment

**LifeCycle method of Servlet**

* **Init**
* **Service**
* **Destroy**

**Init method**

Init method is generally use for initialization of the class object in the servlet, there are two type of init method in the servlet:-

There are two init method in the servlet i.e

1. Init(ServletConfig config):- It is used by web container of the server as there is local variable config is available in the method and if developer try to override the method it will create the problem if we try to fetch the sevletConfig using getServletConfig() in service method.
2. **Void init:-** It is used by the developer for initialization.

So the flow of control in the servlet is like this

* Loading
* Instantiation
* Initialization
  1. init(ServletConfig config){

init()

}

* 1. Init()
* Service
* Destroy

So in the initialization first the flow of control will go to the init(ServletConfig cong) then it will go to the init().

**Why so we need HttpServlet as we have GenericServlet already?**

Ans:- Genric servlet is used for different protocols i.e. http, smtp etc but in the case of httpServlet it is only used for one protocol i.e. http only and generic Servlet don’t contain any get and post method that’s why we need httpServlet method.

**Interface and class of HTTP**

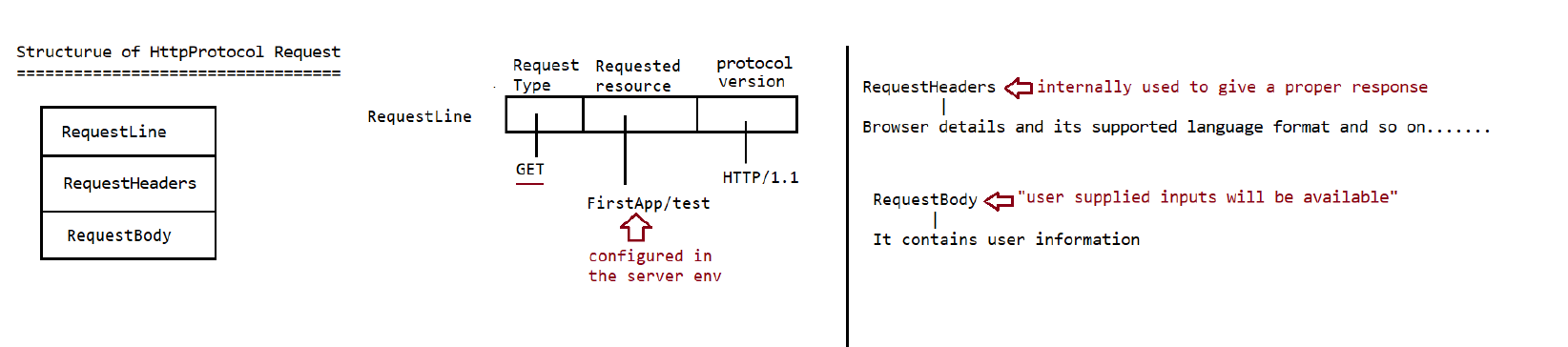
Interface

* HttpServletRequest(I)
* HttpServletResponse(I)
* HttpSession(I)

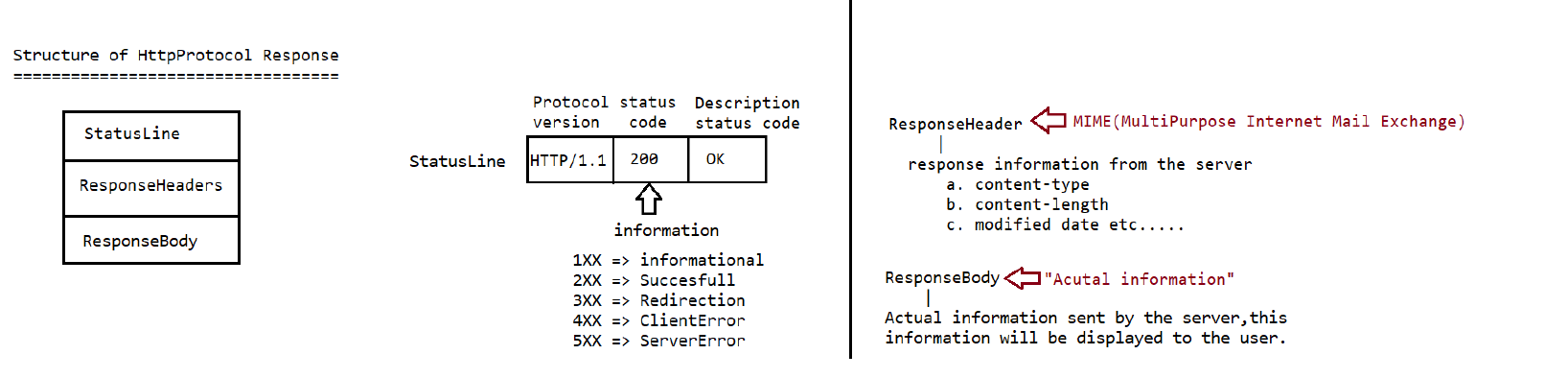
Class

* HttpServlet
* Cookie

Structure of HttpProtocol Request



Structure of HttpProtocol Response



Type of HttpRequest Methods

* Get
* Post
* Head
* Put
* Delete
* Options
* Trace

The first three are introduced in http 1.0 v and the rest was introduced in 1.1v

Requesttype used in webapplications are:-

1. Get (It is also called as idempotent request/safe Request)

🡺 It is used to get the information from server

🡺This request data will be visible in the address bar when we send the request,

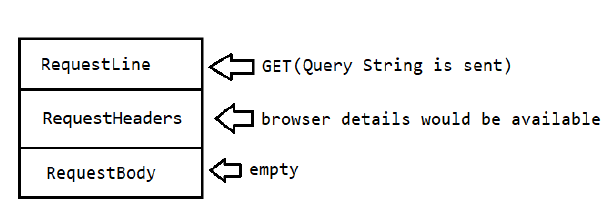
so it is not secured.

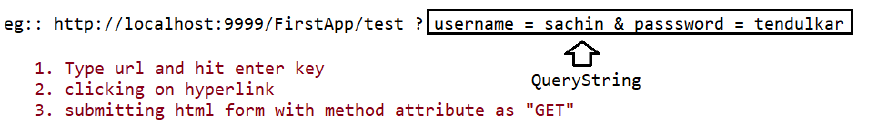
🡺It can be book marked, and it also supports caching of data

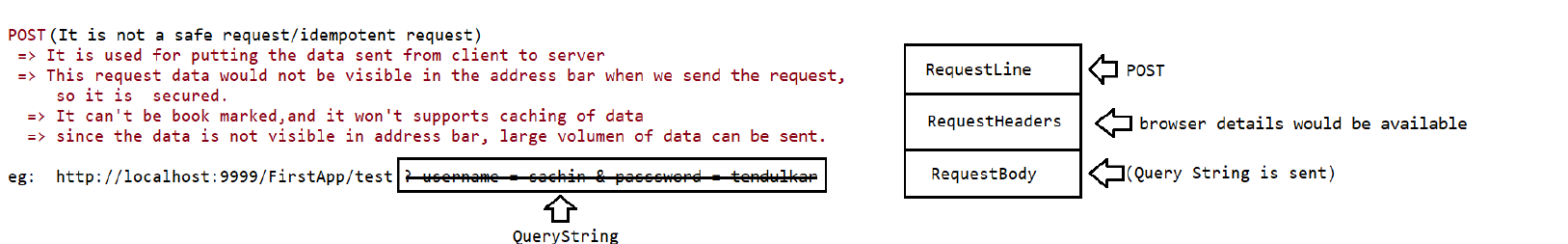
🡺This data is send and visible in address.

🡺This data is send and visible in address bar large volume of data can’ be sent.

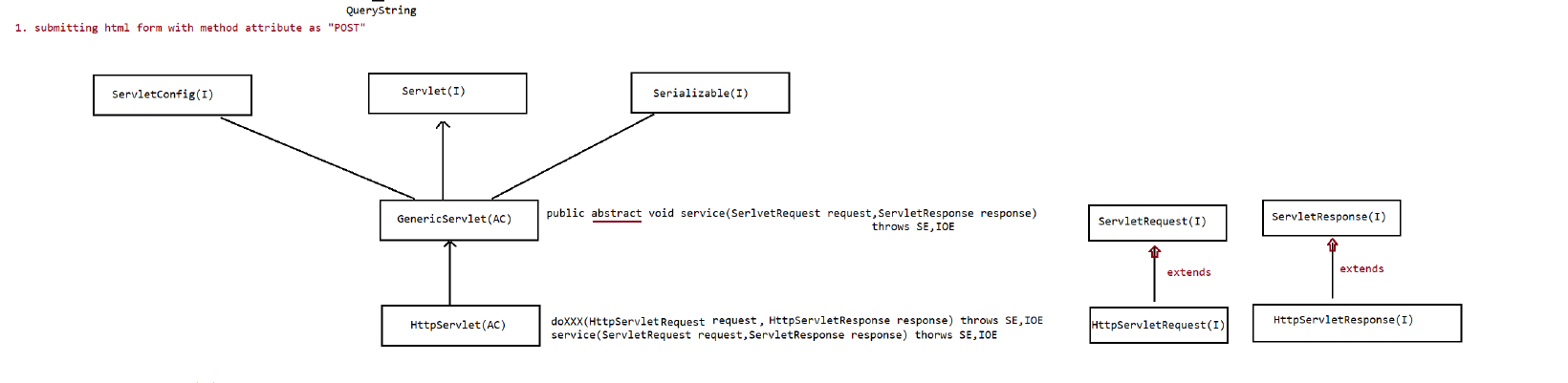
Eg:: <http://locahost:9090/FirstApp/test?username> = Sachin&password=Tendulkar







Inside the **HttpServlet ,** Service method is defined in this way;



First there is servlet service method which type cast HttpServlet service and response and then it will call the HttpServlet service method internally.

**protected void service(SReq req ,SResp resp) throws SE,IOE{**

HttpServletrequest request=(HSR) req;

HttPservletResponse response =(HSR) resp;

Service(request,response);

}

**protected void service(HSR req ,HSR resp) throws SE,IOE{**

**String method = req.getMethod();**

**If(method.equalsIgnoreCase(“POST”)**

**doPost(req,resp);**

**else if(method.equalsIgnoreCase(“GET”)**

**doGet(req,resp);**

**;;;;;;;;**

**;;;;;;;;**

**else**

**return 501 status code saying http method not implemented;**

**}**

As Service method is defined in HttpServlet Abstract class so we don’t we need to define the service method in Httpservlet abstract class.

Let’s see how post method is define in the HttpServlet:-

**protected void doPost(HSR req,HSr resp) throw SE,IOE{**

**return 405 | 400 status code saying http method get status.**

**}**

Now let’s see how the flow of control execute:-

1. **Loading**

Static block

1. **Instantiation**

Constructor(public zero argument)

1. **Initialization**

Init(ServletConfig config)

Init()---meant for Developer method

1. **Request processing**

Public void sevice(sRes res sResp rep)

Protected void service(Httpres res HttpReq req)

1. **Deinstantiation**

Public void destroy();

Q why HttpServlet is abstract class even though all the methods are defined in it?

Ans:- HttpServlet is declared abstract, because there is no sense in using it directly - its methods provide default functionality only (which, in most cases, does nothing). You can add your own functionality by overriding any of those methods in a class that extends HttpServlet.

Let’s see the code here:-

**Log.html:----**

**set path = C:\Program Files\Java\jdk1.8.0\_202\bin;**

**E:**

**set classpath =.;C:\Tomcat 9.0\lib\servlet-api.jar**

**E:**

**cd ..<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Input page</title>**

**</head>**

**<body>**

**<h1>Demostration of Post Method</h1>**

**<form action="./test" method="POST">**

**<table>**

**<tr>**

**<th>Name</th>**

**<td><input type='text' name='username'></td>**

**</tr>**

**<tr>**

**<th>Password</th>**

**<td><input type='password' name='password'></td>**

**</tr>**

**<tr>**

**<th>Login</th>**

**<td><input type='submit' value='login' /></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**Now let’s see the java code:--**

**package in.neuron.controller;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import javax.servlet.ServletConfig;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**/\*\***

**\* Servlet implementation class SecondServlet**

**\*/**

**@WebServlet("/test")**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**static {**

**}**

**public SecondServlet() {**

**System.out.println("Servlet Instantiation");**

**}**

**@Override**

**public void init() throws ServletException {**

**// TODO Auto-generated method stub**

**super.init();**

**ServletConfig config =getServletConfig();**

**System.out.println("Config available obj"+ config);**

**}**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**response.setContentType("text/html");**

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

**out.println("<html><head><title> Output</title></head>");**

**out.println("<body>");**

**out.println("<h1>Getting the response for getrequest Type </h1>");**

**out.println("</body>");**

**out.println("</html>");**

**out.close();**

**}**

**public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

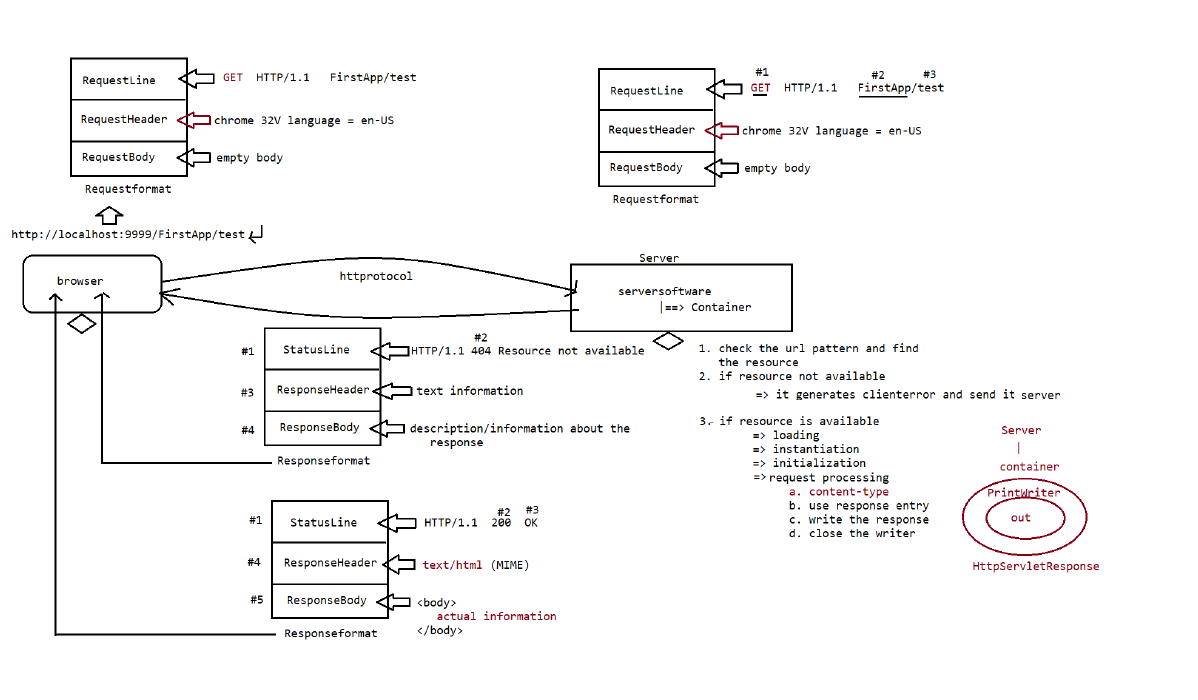
**// TODO Auto-generated method stub**

**doGet(request, response);**

**}**

**}**

**Working of servlet operations:-**



**Let’s see another Project to understand how to fetch the data of request value:-**

To fetch the Request value from the request there is method called getParameter: -

Html pages will be :-

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body bgcolor="green" text="white">

<center>

<h1>Ineuron Integellience private limited:Registration for Student</h1>

</center>

<form method="POST" action="./Reg">

<table align ='center' border =1>

<tr>

<th>Name:-</th>

<td><input type='text' name='username' placeholder='Name' />

</td>

</tr>

<tr>

<th>Contact Number:-</th>

<td><input type='text' name='contacNumber'

placeholder='Contact Number' /></td>

<tr>

<th>E-mail:-</th>

<td><input type='text' name='emailId' placeholder='E-mail id' />

</td>

</tr>

<tr>

<th>Course Details</th>

<td><select name='course' id='courseDetails' multiple ='multiple">

<option value='advanceJava'>Advance java</option>

<option value='hibernate'>Hibernate</option>

<option value='spring'>Spring</option>

<option value='MicrosErvices'>MicroServices</option>

</select>

</td>

</tr>

<tr>

<th></th>

<td>

<input type='Submit' value='register'/>

</td>

</tr>

</table>

</form>

</body>

</html>

Servlet Page:-

package in.neuron.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = "/Reg", loadOnStartup = 1)

public class RegisterServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("Servlet Loading");

}

public RegisterServlet() {

super();

System.out.println("Servlet Instantiation");

}

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

super.init();

System.out.println("servlet Initialization");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

System.out.println("request processing for GetTypr Metod....");

System.out.println("request object Implementation class" +request.getClass().getName());

System.out.println("Response object Implementation class" +response.getClass().getName());

//getting the parameter from the client using html pages

String username = request.getParameter("username");

String contaceNumber =request.getParameter("contacNumber");

String mailId =request.getParameter("emailId");

//getting multipleparameters from the client using html pages

String[] courses =request.getParameterValues("course");

response.setContentType("text/html");

PrintWriter out =response.getWriter();

out.println("<html><head><title> Output</title></head>");

out.println("<body>");

out.println("<table border ='1'align ='center'>");

out.println("<tr><th>Name:- </th><td>"+username+"</td></tr>");

out.println("<tr><th>Number:- </th><td>"+contaceNumber+"</td></tr>");

out.println("<tr><th>e-Mailid:- </th><td>"+mailId+"</td></tr>");

out.println("<tr><th>course Selected </th>");

for(String course:courses) {

out.print("<td>"+course);

}

out.println("</td>");

out.println("</tr>");

out.println("</table>");

out.println("</body>");

out.println("</html>");

out.close();

}

@Override

public void destroy() {

// TODO Auto-generated method stub

super.destroy();

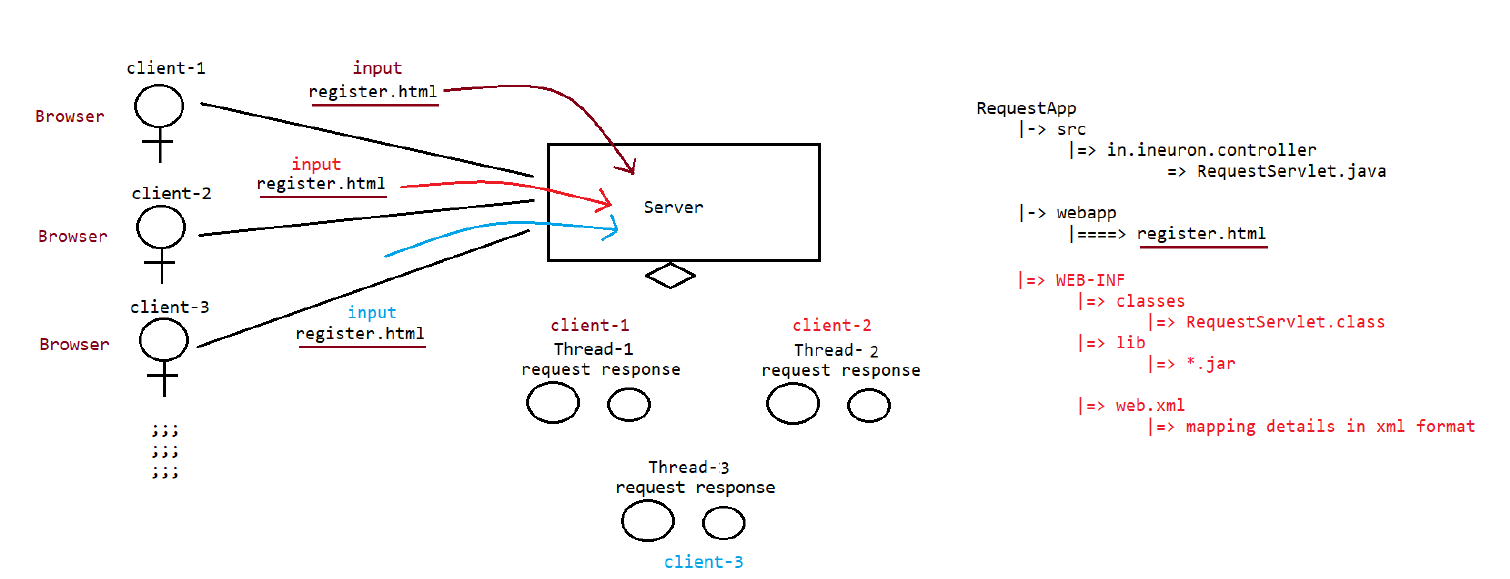
System.out.println("Servlet DeInstantiation");

}

}

Now suppose we have two or more client who are using this servlet, their request are different so, Is response will be same for each client ?

The answer is no, server will provide different object/thread for each client so the response will be different for different client according to their request.;-



As we know that request has three component i.e.

1. Requestline:- It contains what types of request is being given the client, httpversion
2. RequestHeader:-it contains the browser details
3. RequestBody:-It contains the body that client provided.

Let’s see how the browser is using the request header using servlet through code and what type of detail is contained inside the request Header:-

**package in.neuron.controller;**

**import java.io.IOException;**

**import java.io.PrintWriter;**

**import java.util.Enumeration;**

**import javax.servlet.ServletException;**

**import javax.servlet.annotation.WebServlet;**

**import javax.servlet.http.HttpServlet;**

**import javax.servlet.http.HttpServletRequest;**

**import javax.servlet.http.HttpServletResponse;**

**@WebServlet(urlPatterns ="/test")**

**public class BrowserResponserApp extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**static {**

**System.out.println("Servlet Loading......");**

**}**

**public BrowserResponserApp() {**

**super();**

**System.out.println("Server Instatntiation");**

**}**

**@Override**

**public void init() throws ServletException {**

**System.out.println("Servlet Inititialisation");**

**}**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**System.out.println("RequestProcessing for GET Request type.....");**

**response.setContentType("text/html");**

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

**out.println("<html>");**

**out.println("<body>");**

**out.println("<h1>Request Header information.....</h1>");**

**Enumeration<String> headerNames=request.getHeaderNames();**

**out.println("<table border ='1' align ='center'>");**

**out.println("<tr><th>HeaderName</th><th>HeaderValue</th></tr>");**

**while(headerNames.hasMoreElements()) {**

**out.println("<tr>");**

**String headerName =headerNames.nextElement();**

**String headerValue =request.getHeader(headerName);**

**out.println("<td>"+headerName+"</td>");**

**out.print("<td>"+headerValue+"</td>");**

**out.println("</tr>");**

**}**

**out.println("</table>");**

**out.println("</body>");**

**out.println("</html>");**

**}**

**@Override**

**public void destroy() {**

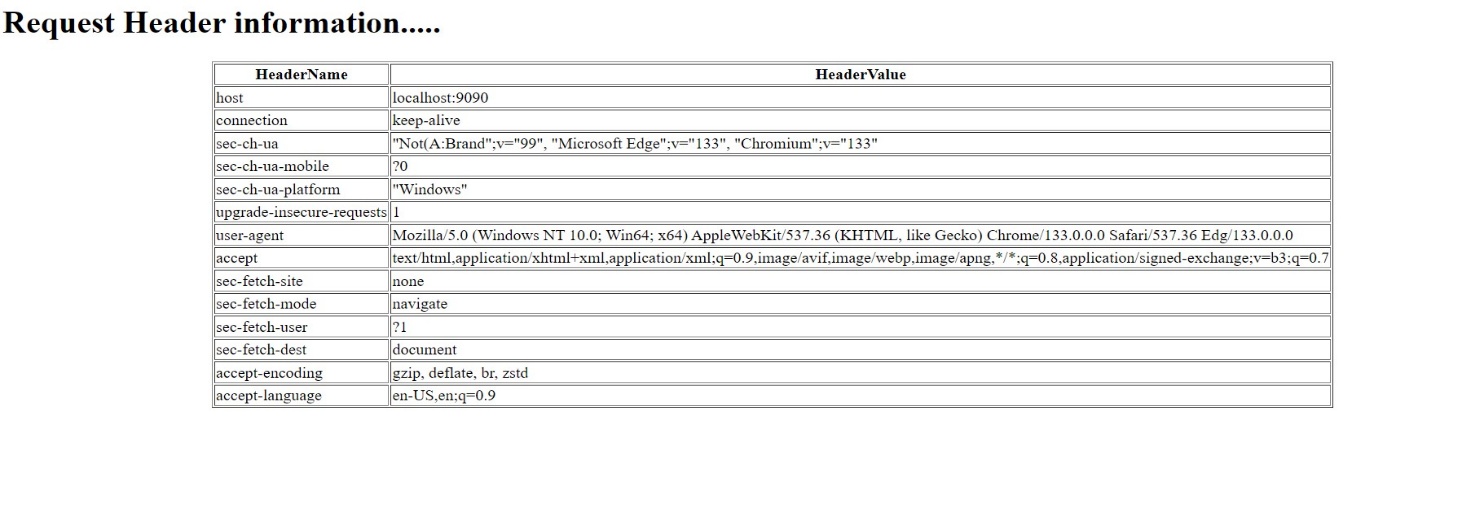
**// TODO Auto-generated method stub**

**super.destroy();**

**System.out.println("Servlet desinstantiation");**

**}**

**}**

****

Output of the above code

So these are the details, which any RequestHeader contains whenever any client request from the server.

Methods which are involved in HttpServletRequest

1. public String getParameter()
2. public String[] getParameterValues()
3. public Enumeration<String> getHeaderNames()
4. public object getHeader(String headerName)
5. public string getRemoteHost()
6. public String getRemoteAddr()
7. public String getRemotePost()
8. public String getServerName()
9. public String getServerPort()

**Note:- Load-on-startup**

1. value should be 0 and any positive number
2. If 2 servlet has same value then we can’t predict the order
3. Less the load-on-startup for a value, that particular servlet loading, instantiation,initializiation will happen.

**@WebServlet(urlPatterns = { "/Demo123" }, loadOnStartup = 1)**

**public class DemoServletApp extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**static {**

**System.out.println("DemoServlet Loading....");**

**}**

**public DemoServletApp() {**

**System.out.println("DemoServlet Instantiation......");**

**}**

**@Override**

**public void init() throws ServletException {**

**System.out.println("DemoServlet Initialization.....");**

**}**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**System.out.println("RequestProcessing for GET Request type.....");**

**System.out.println("Client IPAddress is :: "+request.getRemoteAddr());**

**System.out.println("Client HostDetail is :: "+request.getRemoteHost());**

**System.out.println("Client Port is :: "+request.getRemotePort());**

**System.out.println("ServerName is :: "+request.getServerName());**

**System.out.println("ServerPort is :: "+request.getServerPort());**

**response.setContentType("text/html");**

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

**}**

**}**

**HttpServletResponse methods available**

1. public void setContentType(String name)
2. public PrintWriter getWriter()
3. public ServletOutputStream getOutputStream()

**Note:-Writing image,pdf’s ……as a response**

response.setContentType(“image/jpg”);;;

;;;;;;

ServletOutputStrem os =response.getOutputStream();

Let’s see how to publish the code using servlet:-

package in.ineuron.controller;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class ImageResponseApp

\*/

@WebServlet(urlPatterns={"/Image"})

public class ImageResponseApp extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("servlet loading");

}

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

super.init();

System.out.println("Servlet initialisation");

}

public ImageResponseApp() {

super();

System.out.println("Server Instantilization");

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.setContentType("image/jpg");

ServletOutputStream os = response.getOutputStream();

String path =getServletContext().getRealPath("IMG20230422124757.jpg");

File f = new File(path);

FileInputStream fis =new FileInputStream(f);

byte[] b = new byte[(int)f.length()];

fis.read(b);//reading and placing the image data into byte array.

os.write(b);

os.flush();

os.close();

os.write(null);

os.close();

}

}

Through this code , we can publish the image in the browser , to deploy the image in the browser using servlet we use ServletOutputStream which print the image in the browser using servlet.

Now suppose if we deploy the application in the clound at that time we need to keep the BLOB or Clob in the Soft deployement area of the eclipse and we need to use one of the method i.e getServletContext.getRealPath(“File.name”);

The code will be in suchway :-

String path =getServletContext().getRealPath("IMG20230422124757.jpg");

File f = new File(path);

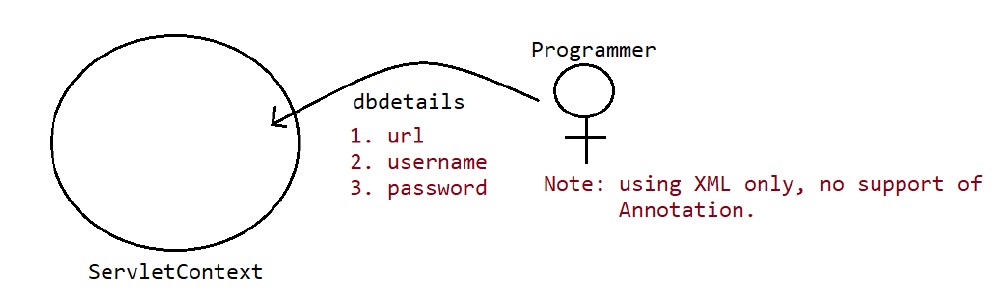
**ServletContext(Interface)**

🡺When we do deployment(manually),server will scan “Webapp” folder and identifies the projects whichis deployed.

🡺All the identified Project will be kept in Meta-space of Server.

🡺For every project which is deployed in Meta-space automatically, “ServletContext” object will be created.

Context object can be accessed in the entire project,meaning in all the servlets of the project.



Q:- How we are going to keep the context object in the servletContext?

Note:- Using XML only, no support of Annotation.

Through this way we can keep servletContext using webxml

**<?xml version="1.0" encoding="UTF-8"?>**

**<web-app xmlns:xsi=**[**http://www.w3.org/2001/XMLSchema-instance**](http://www.w3.org/2001/XMLSchema-instance)

**xmlns="http://java.sun.com/xml/ns/javaee"**

**xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**

**id="WebApp\_ID" version="2.5">**

**<display-name>ServletContextApp</display-name>**

**<context-param>**

**<param-name>jdbcURL</param-name>**

**<param-value>jdbc:mysql://localhost:3306/ineuron</param-value>**

**</context-param>**

**<context-param>**

**<param-name>user</param-name>**

**<param-value>root</param-value>**

**</context-param>**

**<context-param>**

**<param-name>password</param-name>**

**<param-value>Gaurav45</param-value>**

**</context-param>**

**<servlet>**

**<description></description>**

**<display-name>TestServlet</display-name>**

**<servlet-name>TestServlet</servlet-name>**

**<servlet-class>in.neuron.controller.TestServlet</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>TestServlet</servlet-name>**

**<url-pattern>/TestServlet</url-pattern>**

**</servlet-mapping>**

**</web-app>**

Method of servletContext:-

1. **public Enumeration getinitParameterNamer()**
2. **public String getinitParameter(String name)**

**Note:-** In ServletContext object, we can add parameter data from xml as well as attribute data during execution.

public class TestServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("servlet Loading.....");

}

public TestServlet() {

System.out.println("servlet instantiation.....");

}

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

System.out.println("servlet initialization.......");

}

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

ServletContext context =getServletContext();

//attribute data

context.setAttribute("ineuron", "NavinReddy");

response.setContentType("text/html");

PrintWriter out =response.getWriter();

out.println("<html><head><title>Output</title></head>");

out.println("<body>");

out.println("<body align ='centre'>");

out.println("<table border ='1' align='center'>");

out.println("<tr><th>ParameterName</th><th>ParameterValue<th></tr>");

Enumeration<String>initParameterNames =context.getInitParameterNames();

while(initParameterNames.hasMoreElements()) {

String parmeterName =(String) initParameterNames.nextElement();

String parameterValue =context.getInitParameter(parmeterName);

out.println("<tr>");

out.print("<td>"+parmeterName+"</td><td>"+parameterValue+"</td>");

out.println("</tr>");

}

out.println("</table>");

out.println("</body>");

out.println("</html>");

}

}

**ServletConfig:-**

🡺This object is used to tore the configuration details of a particular servlet like logical name of the servlet,initialization parameters, and so on…..

🡺Using ServletConfig we will get toknow the complete view of a particular servlet.

🡺Loading 🡪Static Block

Instantiation🡪public zero argument constructor

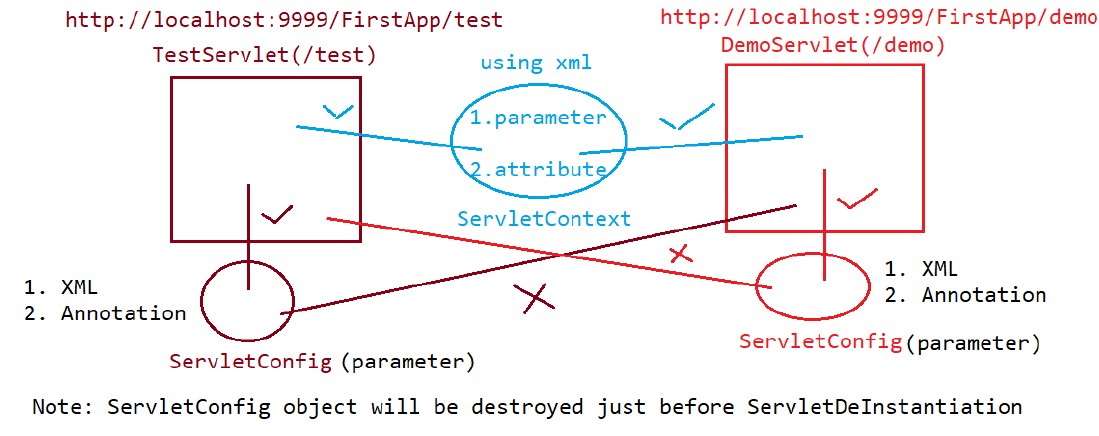
Initialization🡪public void init(ServletConfig config)

RequestProcessing🡪public void doXXX(HSR request,HSR response) throw SE,IOE

Deinstantiation🡪public void destroy()

🡺 From the above we can say that servletConfig is automatically get created and injected after instantiation as **init method** has **servletConfig** as parameter.

🡺Every servlet has it’s own ServletConfig as every servlet load ,instantiate,initialize ,process the request and deinstantiate.



**Q. In how many ways we can save the data in the config?**

**Ans:-1.XML**

**2.Annotation**

And in ServletConfig we can only store parameterized data.

|  |  |
| --- | --- |
| **ServletConfig** | **ServletContext** |
| 1.ServletConfig is different for different servlet | 1.Servlet Context is same for whole project |
| 2.We can store servlet using xml and annotation both | 2.We can store servletcontext only by xml. |
| 3.Only parameterized data is stored in servlet config | 3.Parameterized and attribute data are stored in servlet context. |
| 4. ServletConfig object is obtained by getServletConfig() method. | 4. ServletContext object is obtained by getServletContext() method. |
|  |  |
|  |  |

Methods used in servlet config

1.public Enumeration getInitParameterNames()

2.public string getinitparameter(String name)

We Can store servletconfig using xml , so where we need to those parameter inside the servlet. As we know that every servlet has it’s own servletConfig so we need to keep that servletConfig inside the class level of xml. Like this.

**<servlet>**

**<servlet-name>ServletConfigApp</servlet-name>**

**<servlet-class>in.ineuron.controller.ServletConfigApp</servlet-class>**

**<init-param>**

**<param-name>jdbcUrl</param-name>**

**<param-value>jdbc:mySql:///ineuron</param-value>**

**</init-param>**

**</servlet>**

The code for the servlet config will be:-

/\*\*

\* Servlet implementation class ServletConfigApp(/ServletConfigApp)

\*/

public class ServletConfigApp extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("servlet loading");

}

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

System.out.println("servlet Instatantiation");

}

public ServletConfigApp() {

System.out.println("servlet Initialization");

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

System.out.println("Request processing for GET request type......");

response.setContentType("text/html");

PrintWriter out =response.getWriter();

out.println("<html><head><title>Output</title></head>");

out.println("<body>");

out.println("<body align ='center'>");

out.println("<table border ='1'>");

out.println("<tr><th>ParameterName</th><th>ParameterValue<th></tr>");

ServletConfig config=getServletConfig();

Enumeration<String>initParameterNames =config.getInitParameterNames();

while(initParameterNames.hasMoreElements()) {

String parmeterName =(String) initParameterNames.nextElement();

String parameterValue =config.getInitParameter(parmeterName);

out.println("<tr>");

out.print("<td>"+parmeterName+"</td><td>"+parameterValue+"</td>");

out.println("</tr>");

}

out.println("</table>");

out.println("</body>");

out.println("</html>");

}

}

The code for the other servlet program will be:-

/\*\*

\* Servlet implementation class DemoServletConfig(/DemoServletConfig)

\*/

public class DemoServletConfig extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("servlet loading");

}

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

System.out.println("servlet Instatantiation");

}

public DemoServletConfig() {

System.out.println("servlet Initialisation");

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

System.out.println("Request processing for GET request type......");

response.setContentType("text/html");

PrintWriter out =response.getWriter();

out.println("<html><head><title>Output</title></head>");

out.println("<body>");

out.println("<body align ='center'>");

out.println("<table border ='1'>");

out.println("<tr><th>ParameterName</th><th>ParameterValue<th></tr>");

ServletConfig config=getServletConfig();

Enumeration<String>initParameterNames =config.getInitParameterNames();

while(initParameterNames.hasMoreElements()) {

String parmeterName =(String) initParameterNames.nextElement();

String parameterValue =config.getInitParameter(parmeterName);

out.println("<tr>");

out.print("<td>"+parmeterName+"</td><td>"+parameterValue+"</td>");

out.println("</tr>");

}

out.println("</table>");

out.println("</body>");

out.println("</html>");

}

}

The xml code will be like:-

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5">

<display-name>ServletConfigUsingXml</display-name>

<servlet>

<servlet-name>ServletConfigApp</servlet-name>

<servlet-class>in.ineuron.controller.ServletConfigApp</servlet-class>

<init-param>

<param-name>jdbcUrl</param-name>

<param-value>jdbc:mySql:///ineuron</param-value>

</init-param>

<init-param>

<param-name>username</param-name>

<param-value>root</param-value>

</init-param>

<init-param>

<param-name>password</param-name>

<param-value>Gaurav45</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>ServletConfigApp</servlet-name>

<url-pattern>/ServletConfigApp</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>DemoServletConfig</servlet-name>

<servlet-class>in.neuron.controller.DemoServletConfig</servlet-class>

<init-param>

<param-name>Company</param-name>

<param-value>Oracle</param-value>

</init-param>

<init-param>

<param-name>courseName</param-name>

<param-value>JavaFullStack</param-value>

</init-param>

<init-param>

<param-name>Mentor</param-name>

<param-value>NaveenReddy</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>DemoServletConfig</servlet-name>

<url-pattern>/DemoServletConfig</url-pattern>

</servlet-mapping>

</web-app>

ServletConfig using Annotation:-

@WebServlet(

urlPatterns = { "/ServletConfigWithAnnotation" },

initParams = {

@WebInitParam(name = "jdbcUrl", value = "jdbc:mysql:///ineuron"),

@WebInitParam(name = "user", value = "root"),

@WebInitParam(name = "password", value = "Gaurav45")

})

The code for the servlet will be like:-

/\*\*

\* Servlet implementation class ServletConfigWithAnnotation

\*/

@WebServlet(

urlPatterns = { "/ServletConfigWithAnnotation" },

initParams = {

@WebInitParam(name = "jdbcUrl", value = "jdbc:mysql:///ineuron"),

@WebInitParam(name = "user", value = "root"),

@WebInitParam(name = "password", value = "Gaurav45")

})

public class ServletConfigWithAnnotation extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("servlet loading");

}

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

System.out.println("servlet Instatantiation");

}

public ServletConfigWithAnnotation() {

System.out.println("servlet Initialization");

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

System.out.println("Request processing for GET request type......");

response.setContentType("text/html");

PrintWriter out =response.getWriter();

out.println("<html><head><title>Output</title></head>");

out.println("<body>");

out.println("<body align ='center'>");

out.println("<table border ='1'>");

out.println("<tr><th>ParameterName</th><th>ParameterValue<th></tr>");

ServletConfig config=getServletConfig();

Enumeration<String>initParameterNames =config.getInitParameterNames();

while(initParameterNames.hasMoreElements()) {

String parmeterName =(String) initParameterNames.nextElement();

String parameterValue =config.getInitParameter(parmeterName);

out.println("<tr>");

out.print("<td>"+parmeterName+"</td><td>"+parameterValue+"</td>");

out.println("</tr>");

}

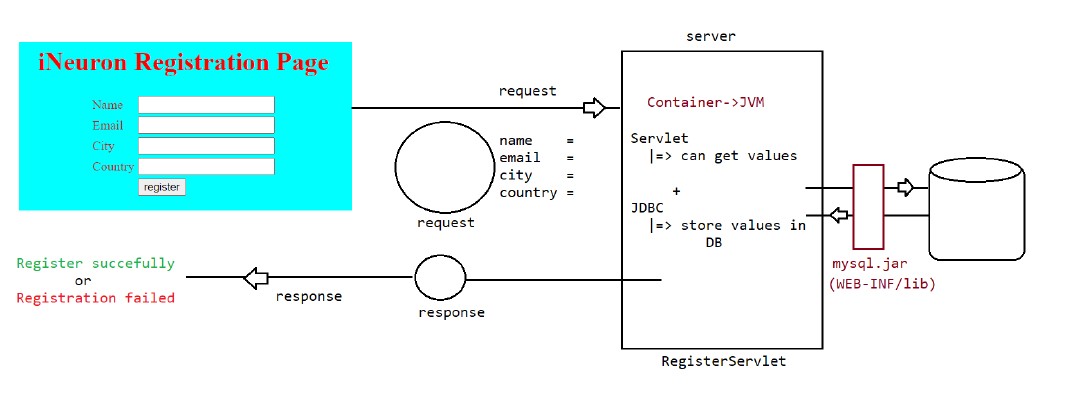
out.println("</table>");

out.println("</body>");

out.println("</html>");

}

}



**Connecting Servlet with the JDBC:- Html File**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Registration Page</title>

</head>

<body bgcolor ="skyblue" text="red">

<h1 style='color: red; text-align: center'>iNeuron Registration

page</h1>

<form method ="post" action="./reg">

<table align='center'>

<tr>

<td>Name</td>

<td><input type='text' name ='username'placeholder='username'/></td>

</tr>

<tr>

<td>Email</td>

<td><input type='email' name ='email'placeholder='email'/></td>

</tr>

<tr>

<td>City</td>

<td><input type='text' name ='city'placeholder='city'/></td>

</tr>

<tr>

<td>Country</td>

<td><input type='text' name ='country'placeholder='Country'/></td>

</tr>

<tr>

<td><input type='submit' value='Register'/></td>

</tr>

</table>

</form>

</body>

</html>

Servlet code for connectionin the database:-

@WebServlet(urlPatterns = { "/reg" }, initParams = {

@WebInitParam(name = "url", value = "jdbc:mysql://localhost:3306/ineuron"),

@WebInitParam(name = "user", value = "root"), @WebInitParam(name = "password", value = "Gaurav45") })

public class RegisterDb extends HttpServlet {

private static final long serialVersionUID = 1L;

static {

System.out.println("RegisterServlet class is loading...");

}

public RegisterDb() {

System.out.println("Register Servlet class is instantiated...");

}

private Connection conn = null;

@Override

public void init() throws ServletException {

// TODO Auto-generated method stub

ServletConfig config = getServletConfig();

String url = config.getInitParameter("url");

String password = config.getInitParameter("password");

String user = config.getInitParameter("user");

try {

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

conn = DriverManager.getConnection(url, user, password);

} catch (SQLException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// TODO Auto-generated method stub

String username = request.getParameter("username");

String email = request.getParameter("email");

String city = request.getParameter("city");

String Country = request.getParameter("country");

int rowcount = 0;

PreparedStatement pstmt = null;

String insertQuery = "INSERT INTO people (username, email, city, country) VALUES (?, ?, ?, ?);";

try {

if (conn != null) {

pstmt = conn.prepareStatement(insertQuery);

if (pstmt != null) {

pstmt.setString(1, username);

pstmt.setString(2, email);

pstmt.setString(3, city);

pstmt.setString(4, Country);

rowcount = pstmt.executeUpdate();

}

}

} catch (SQLException sql) {

sql.printStackTrace();

} catch (Exception Ex) {

Ex.printStackTrace();

}

String username1 = null;

String email1 = null;

String city1 = null;

String country1 = null;

ResultSet rs = null;

PreparedStatement pstmt1 = null;

int peopleId = 1;

String selectQuery = "select username,email,city,country from people where peopleid=?;";

try {

if (conn != null) {

pstmt1 = conn.prepareStatement(selectQuery);

if (pstmt1 != null) {

pstmt1.setInt(1, peopleId);

rs = pstmt1.executeQuery();

if (rs != null) {

while (rs.next()) {

username1 = rs.getString(1);

email1 = rs.getString(2);

city1 = rs.getString(3);

country1 = rs.getString(4);

}

}

}

}

} catch (SQLException sql) {

sql.printStackTrace();

} catch (Exception Ex) {

Ex.printStackTrace();

}

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<html><head><title>Output</title></head>");

out.println("<body bgcolor =\"skyblue\" text=\"red\">");

out.println("<table align ='center'>");

out.println("<tr>");

out.println("<td>Username</td><td>" + username1 + "</td>");

out.println("</tr>");

out.println("<tr>");

out.println("<td>Email</td><td>" + email1 + "</td>");

out.println("</tr>");

out.println("<tr>");

out.println("<td>City</td><td>" + city1 + "</td>");

out.println("</tr>");

out.println("<tr>");

out.println("<td>Country</td><td>" + country1 + "</td>");

out.println("</tr>");

out.println("</table>");

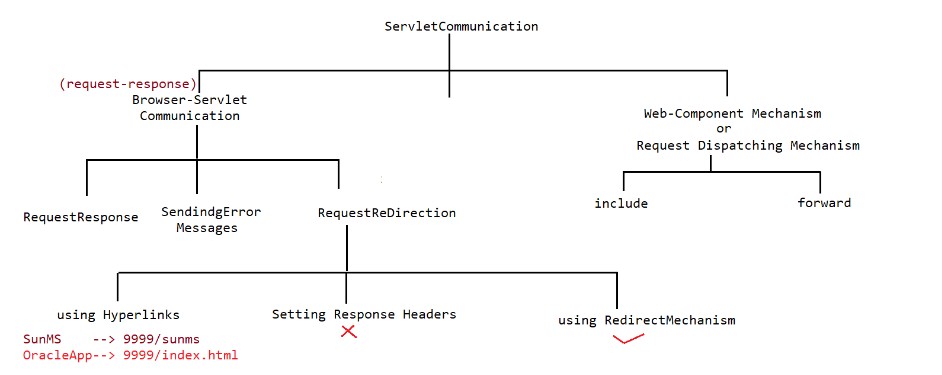
out.println("</body>");

out.println("</html>");

}

}

**Servlet Communication**



**Browser-Servlet Communicaton**:- Till now whatever application we have created in which request is given by the browser and response is given back to the browser by the servlet is called browser-Servlet Communication. In this communication , request-response communication happens.

**SendingErrorMessages**:- Let’s see how to build sending error message code in the browser.

Now suppose we want to send error to the user if any condition dono’t match then how we are going to send the error to the user using servlet.

The Answer is simple response object has one method i.e.

* SendError(“Error message”)
* SendError(int errorCode, “Error message”)

Error code are

1XX-199-> Information Redirectio

2XX-299->Success

3XX-399->Redirection

4XX-499->Client Error

5XX-599->Server Error

Thorough these methods of response object we can send error in servlet like this:-

response.sendError(504, "user age is not sufficient for this Recruitment");

Now let’s code to Send Error message:-

**Requirement:-** Here we need to create a html page where input’s are name,age,email,mobile , so we need to check if age is greater that 18 and less than 30 then only canidate is eligible for recruitment. Let’s see the html code:-

Html page will be

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Registration Page</title>**

**</head>**

**<body>**

**<FORM color='red'>**

**<H2>iNeuron consultancy Services</H2>**

**<h3>User Registration Page</h3>**

**</FORM>**

**<form method='post' action="./reg">**

**<table>**

**<tr>**

**<td>Name</td>**

**<td><Input type='text' name='name' placeholder='name' /></td>**

**</tr>**

**<tr>**

**<td>Age</td>**

**<td><Input type='text' name='uage' placeholder='age' /></td>**

**</tr>**

**<tr>**

**<td>Email</td>**

**<td><Input type='email' name='email' placeholder='email' /></td>**

**</tr>**

**<tr>**

**<td>Mobile</td>**

**<td><Input type='text' name='mobile' /></td>**

**</tr>**

**<tr>**

**<td><Input type='submit' value='Register' /></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

The code for java will be :-

**@WebServlet(urlPatterns = {"/reg"},**

**initParams = {**

**@WebInitParam(name = "url", value = "jdbc:mysql://localhost:3306/userdatabase"),**

**@WebInitParam(name = "user", value = "root"),**

**@WebInitParam(name = "password", value = "Gaurav45")})**

**public class RegistrationServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**private Connection conn = null;**

**public RegistrationServlet() {**

**super();**

**// TODO Auto-generated constructor stub**

**}**

**@Override**

**public void init() throws ServletException {**

**// TODO Auto-generated method stub**

**ServletConfig config = getServletConfig();**

**String url = config.getInitParameter("url");**

**String user = config.getInitParameter("user");**

**String psw = config.getInitParameter("password");**

**try {**

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

**conn = DriverManager.getConnection(url, user, psw);**

**} catch (SQLException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**} catch (ClassNotFoundException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**protected void doPost(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**String name = request.getParameter("name");**

**String age = request.getParameter("uage");**

**String email = request.getParameter("email");**

**String mobile = request.getParameter("mobile");**

**PreparedStatement pstmt = null;**

**PreparedStatement pstmt1 = null;**

**ResultSet rs = null;**

**int rowCount = 0;**

**String dbAge = null;**

**String dbName = null;**

**String dbmail = null;**

**String dbMobile = null;**

**int userId = 3;**

**int parseAge = 0;**

**if (age != null) {**

**parseAge = Integer.parseInt(age);**

**}**

**if (parseAge < 18 || parseAge > 30) {**

**response.sendError(504, "user age is not sufficient for this Recruitment");**

**}**

**else {**

**String insertQuery = "insert into userstable(username,userage,email,mobile) values(?,?,?,?)";**

**try {**

**if (conn != null) {**

**pstmt = conn.prepareStatement(insertQuery);**

**if (pstmt != null) {**

**pstmt.setString(1, name);**

**pstmt.setString(2, age);**

**pstmt.setString(3, email);**

**pstmt.setString(4, mobile);**

**rowCount = pstmt.executeUpdate();**

**System.out.println("rowUpdate" + rowCount);**

**}**

**}**

**} catch (SQLException sq) {**

**sq.printStackTrace();**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**}**

**String selectQuery = "select userage,username,email,mobile from userstable where userid=?";**

**try {**

**if (conn != null) {**

**pstmt1 = conn.prepareStatement(selectQuery);**

**if (pstmt1 != null) {**

**pstmt1.setInt(1, userId);**

**rs = pstmt1.executeQuery();**

**while (rs.next()) {**

**dbAge = rs.getString(1);**

**dbName = rs.getString(2);**

**dbmail = rs.getString(3);**

**dbMobile = rs.getString(4);**

**}**

**}**

**}**

**} catch (SQLException sq) {**

**sq.printStackTrace();**

**} catch (Exception ex) {**

**ex.printStackTrace();**

**}**

**// if age is less than 10 greater than 30 not eligible for recruitment**

**response.setContentType("text/html");**

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

**out.println("<html><head><title>Output</title></head>");**

**out.println("<body bgcolor =\"skyblue\" text=\"red\">");**

**out.println("<font color ='red'>");**

**out.println("<h2>ineuron Consultancy Services</h2>");**

**out.println("<h2>User registration Details</h2>");**

**out.println("</font>");**

**out.println("<table border =1>");**

**out.println("<tr>");**

**out.println("<td>Username</td><td>" + dbName + "</td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<td>User age</td><td>" + dbAge + "</td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<td>Email</td><td>" + dbmail + "</td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<td>Mobile</td><td>" + dbMobile + "</td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<td>Registration status</td><td>Registered Successful</td>");**

**out.println("</tr>");**

**out.println("</table>");**

**out.println("</body>");**

**out.println("</html>");**

**}**

**}**

**}**

The above code will represent the send error message

* **Request Redirection using hyperlink:-**  To redirect using hyperlink we just need to use anchor tag of html to redirect the flow in the servlet in this way.

Now suppose if there is any site which is owned by new owner and the new owner is providing the same facilities in his site so there will be no need to keep two sites parallely . But here is the catch most of the user know the name of old owner website so what new owner will do is they will redirect their site name from the older website . so whenever any user comes to the old web sites , they will redirect it to the new one by clicking into the hyperlink. That practice is called the Request Redirection using hyperlink.

Servlet Code will be like in this way:-

**@WebServlet("/SunMSServlet")**

**public class SunMSServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**response.setContentType("text/html");**

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

**out.println("<html><head><title>Output</title></head>");**

**out.println("<body>");**

**out.println("<h1>To get the services of java click on below link</h1>");**

**out.println("<a href='http://localhost:9090/OracleApp/index.html'>Oracle site</a>");**

**out.println("</body>");**

**out.println("</html>");**

**}**

**}**

For the above servlet code we need to write redirected html in which flow will be directed like in this way:-

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Response Page</title>**

**</head>**

**<body>**

**<h1 style ='color:red; text-align:center'> Welcome to the world of oracle products</h1>**

**<marquee>Oracle had takenover java.......</marquee>**

**</body>**

**</html>**

**Request redirection using RedirectMechanism**

In this we donot need to use hyperlink to direct the servlet , when ever any user search for old link it will automatically redirect it to the new website.

In response object of the servlet there is one method called **sendDirect(String)** which accepts the String, In that string we simply need to type the link of the website and it automatically redirect it to that website.

Let’s see how:-

**String location=”http://localhost:9090/OracleApp/index.html**

**response.sendRedirect(location);**

The servlet code is given below for the same

**@WebServlet("/SunMSServlet")**

**public class SunMSServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

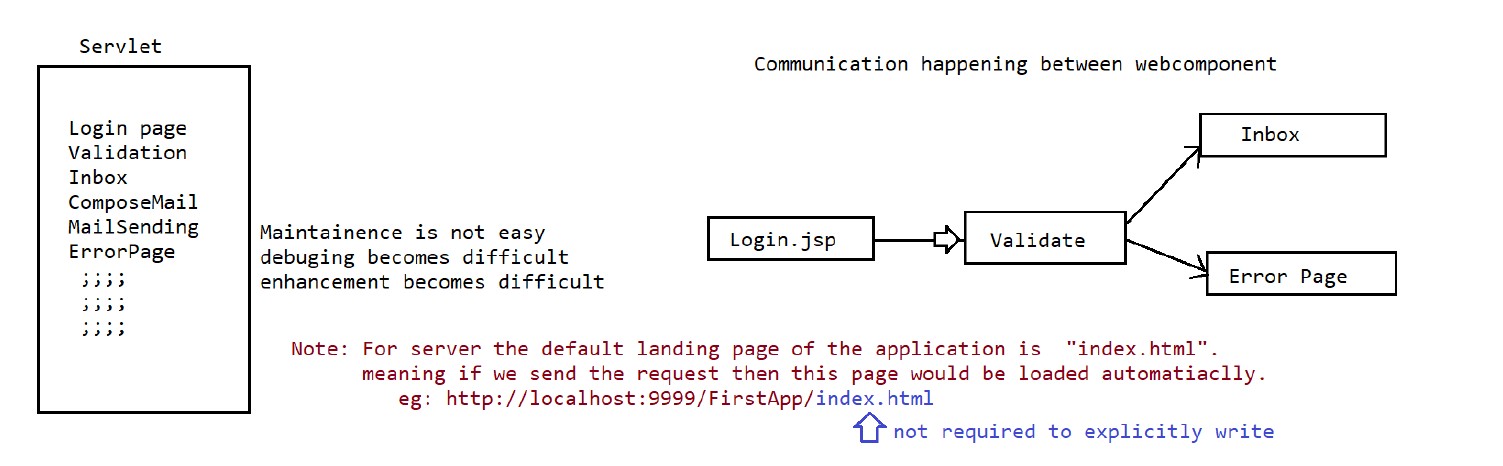
**String location=”http://localhost:9090/OracleApp/index.html**

**response.sendRedirect(location);**

**}**

**}**

**ServletCommunication using Request Dispatch Mechanism(Include):-**



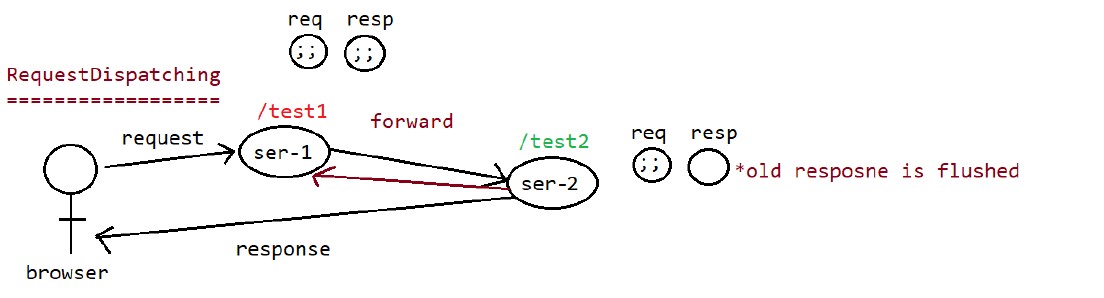
Let’s take an example of Gmail, if we see the above diagram , we can see how we can login the gmail and what are the proceduce to login ,composing the mail , sending the mail and so on.

Now suppose if we write all the features of the gmail in the same servlet , is it easy to maintain that servlet . The answer is no. It is very difficult to manage and some time impossible for enhancement,

So what we do is we simple segregate the different functionality of the servlet in different servlet. And using communication we achieve our requirement.

Let’s see how we are going to do the communication between the different servlet of the application.

Default response of HttpServlet is HTML and text so we need to only write the body part of the html.



Now suppose we have two servlet:-

* FirstServlet
* Seconndservlet

From the diagram , we can see browser gives the request to the **FirstServlet** and then the flow of control will go to the **Secondservlet** and the response will go back to the browser. This is called **request Dispatching.**  Let’s see how to do the request dispatching.

* Returns a [RequestDispatcher](https://javaee.github.io/javaee-spec/javadocs/java.servlet/javax/servlet/RequestDispatcher.html) object that acts as a wrapper for the resource located at the given path. A RequestDispatcher object can be used to forward a request to the resource or to include the resource in a response. The resource can be dynamic or static.

The pathname specified may be relative, although it cannot extend outside the current servlet context. If the path begins with a "/" it is interpreted as relative to the current context root. This method returns null if the servlet container cannot return a RequestDispatcher.

The difference between this method and [ServletContext.getRequestDispatcher(java.lang.String)](https://javaee.github.io/javaee-spec/javadocs/java.servlet/javax/servlet/ServletContext.html#getRequestDispatcher-java.lang.String-) is that this method can take a relative path.

The Request dispatcher can only be used in same project, we can’t use request Dispatcher for other projects

The way to use request dispatcher is

**RequestDispatcher rd =request.getRequestDispatcher(“/test2);**

**Rd.forward(request,response);**

Let’s see the code of the firstServlet:-

This is the first servlet which is redirecting to the second servlet:-

**@WebServlet(urlPatterns={"/test1"})**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**@Override**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

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

**//default response type is "text/html" only**

**out.println("<h1 style ='color:red; text-align:center;'>This is First Servlet</h1>");**

**out.close();**

**RequestDispatcher rd =request.getRequestDispatcher("/test2");**

**rd.forward(request,response);**

**}**

**}**

This is second servlet which is being redirected by the first servlet:-

package in.neuron.controller;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class SecondServlet

\*/

**@WebServlet(urlPatterns={"/test2"})**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

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

**out.println("<h1 style ='color:green; text-align:center;'>This is second Servlet</h1>");**

**out.close();**

**}**

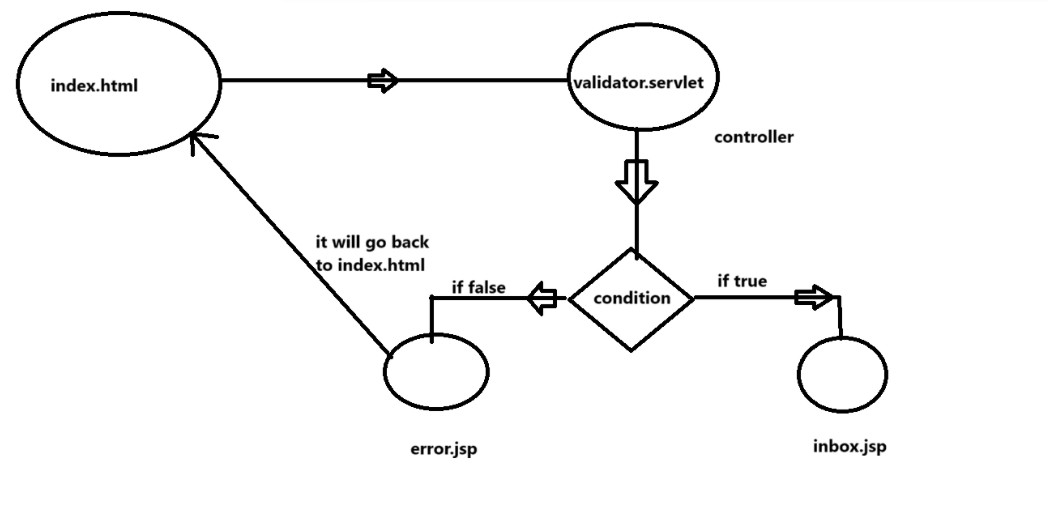
**}**

Other way of RequestDispatcher is:-

**ServletContext context =getServletContext();**

**RequestDispatcher rd =context.getRequestDispatcher("/inbox.jsp");**

**rd.forward(request,response);**



As we can see in the diagram, we need to create an app which has index.html, which contain input type of username and password.

**Let’s see the code of index.html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Login Page</title>**

**</head>**

**<body>**

**<h1>This is forward demo</h1>**

**<form action='./test1 ' method="get">**

**<table>**

**<tr>**

**<td>Name</td>**

**<td><input type="text" name='username' placeholder='username' />**

**</td>**

**</tr>**

**<tr>**

**<td>Password</td>**

**<td><input type="password" name='password'**

**placeholder='password' /><br></td>**

**</tr>**

**<tr>**

**<td><input type ='submit' value='validate'></td>**

**</tr>**

**</table>**

**</form>**

**</body>**

**</html>**

**Now let’s see the code of validator.servlet**

**@WebServlet(urlPatterns={"/test1"})**

**public class validateServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**@Override**

**protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {**

**String uname =req.getParameter("username");**

**String upwd =req.getParameter("password");**

**if(uname.equals("sachin") && upwd.equals("tendulkar")) {**

**//valid page redirect to inbox.jsp**

**ServletContext context =getServletContext();**

**RequestDispatcher rd =context.getRequestDispatcher("/inbox.jsp");**

**rd.forward(req, resp);**

**}**

**else {**

**//invalid page redirect to erro.jsp**

**RequestDispatcher rd =req.getRequestDispatcher("/error.jsp");**

**rd.forward(req, resp);**

**}**

**}**

**}**

**Now let’s see the inbox.jsp**

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Inbox.jsp</title>**

**</head>**

**<body>**

**<h1>**

**<p>This is inbox jsp page</p>**

**</h1>**

**</body>**

**</html>**

**Now let’s see the error.jsp**

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Error Page</title>**

**</head>**

**<body>**

**<h1> This is error page your credential are invalide plz try again</h1>**

**<a gref='./index.html'>login page</a>**

**</body>**

**</html>**

**RequestDispatcher object can be obtained in 2 ways:-**

**ServletContext context =getServletContext();**

**RequestDispatcher rd = context.getRequestDispatcher(“/test2”); //relative path only**

**b.ServletRequest**

**RequestDispatcher rd =request.getRequestDispatcher(“/test2”); //relative path**

**RequestDispatcher rd =request.getRequestDispatcher(“test2”);//absolute path**

Absolute Path 🡺 It indicates we need to write from the root level

Relative Path🡺 It indicates we need to weite from the current directory(./ or /)

Let’s see another example where we set attribute in one servlet and then redirect to the second Servlet

Let’s see the first servlet:-

**package in.ineuron.controller;**

**import java.io.IOException;**

**@WebServlet("/test")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**request.setAttribute("sachin", "cricketer");**

**request.setAttribute("NavinReddy","java");**

**RequestDispatcher rd =request.getRequestDispatcher("/test2");**

**rd.forward(request, response);**

**}**

**}**

**Let’s see the second servlet:-**

**@WebServlet("/test2")**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public SecondServlet() {**

**super();**

**// TODO Auto-generated constructor stub**

**}**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

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

**out.println("<h1>Forward request attributes </h1>");**

**Enumeration<String> attributesNames=request.getAttributeNames();**

**out.println("<table border ='1'>");**

**out.println("<tr><th>Name</th><th>value</th></tr>");**

**while(attributesNames.hasMoreElements()) {**

**out.println("<tr>");**

**String name =(String) attributesNames.nextElement();**

**Object value =request.getAttribute(name);**

**out.println("<th>"+name+"</th>");**

**out.println("<td>"+value+"</td>");**

**out.println("</tr>");**

**}**

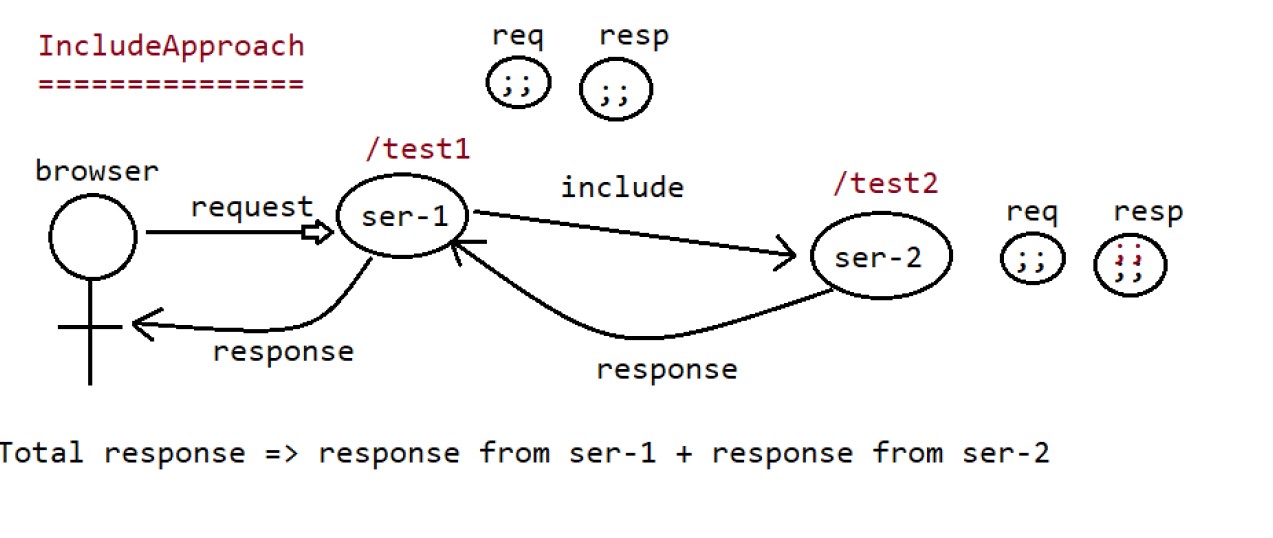
**out.println("</table>");**

**out.close();**

**}**

**}**

**Include Approach**

****

In Include approach, first the flow of control will go to the servlet1 then it will go to the servlet2 and then again comes back to the servlet2 so we can say that in include it just include servlet2 in sevlet1 and response is given back to servlet1.

The code of servlet1 is :-

**@WebServlet("/FirstServlet")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

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

**out.println("<h1> Hello this is First Servlet</h1>");**

**RequestDispatcher rd =request.getRequestDispatcher("/test2");**

**rd.include(request, response);**

**out.println("<h1>Hi, This is First Servlet Again....</h1>");**

**out.close();**

**}**

**}**

The code of servlet2 is:-

**@WebServlet("/test2")**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**// TODO Auto-generated method stub**

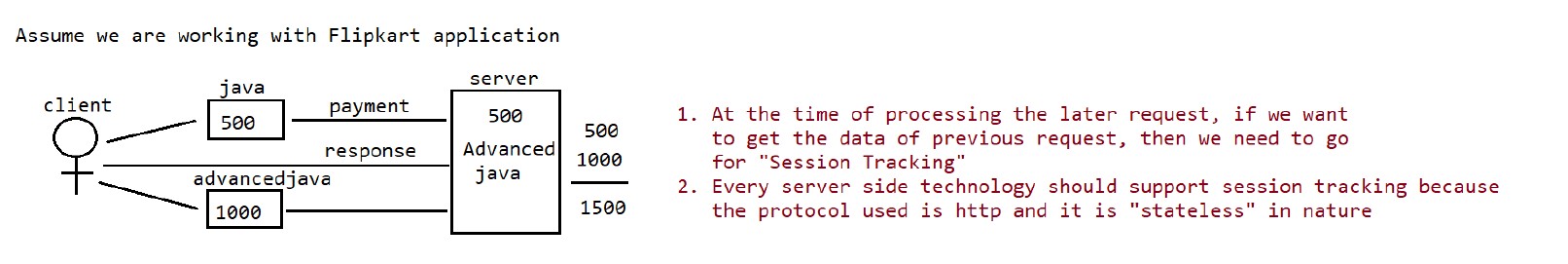
**PrintWriter out = response.getWriter();**

**out.println("<h1>This is second Servlet.....</h1>");**

**}**

**}**

Assume we are purchasing core java book from flipkart, now we are the at the cart section where we need to do pay, suddenly we saw advance java in our suggestion and want to purchase that book also but flipkart is coded using servlet which is using http protocol which is stateless in nature so when we try to add the another book in the cart and previous book get deleted.



We have four ways of tracking Session.

a.HttpSession Tracking mechanism

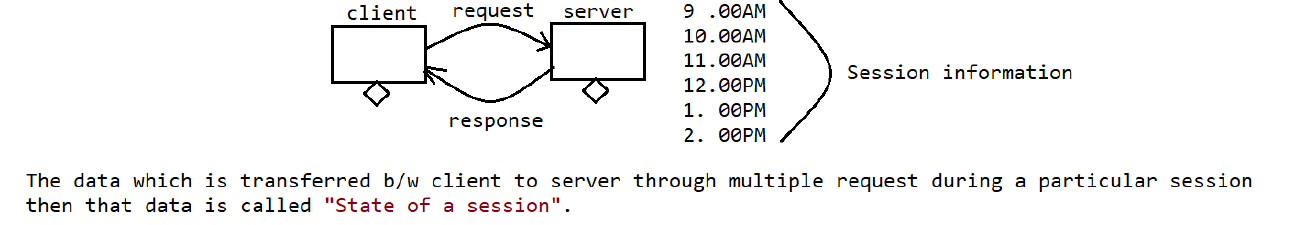
b.Cookies Session Tracking mechanism

c.URL-Rewriting Session Tracking mechanism

d.Hidden formField session Tracking mechanism

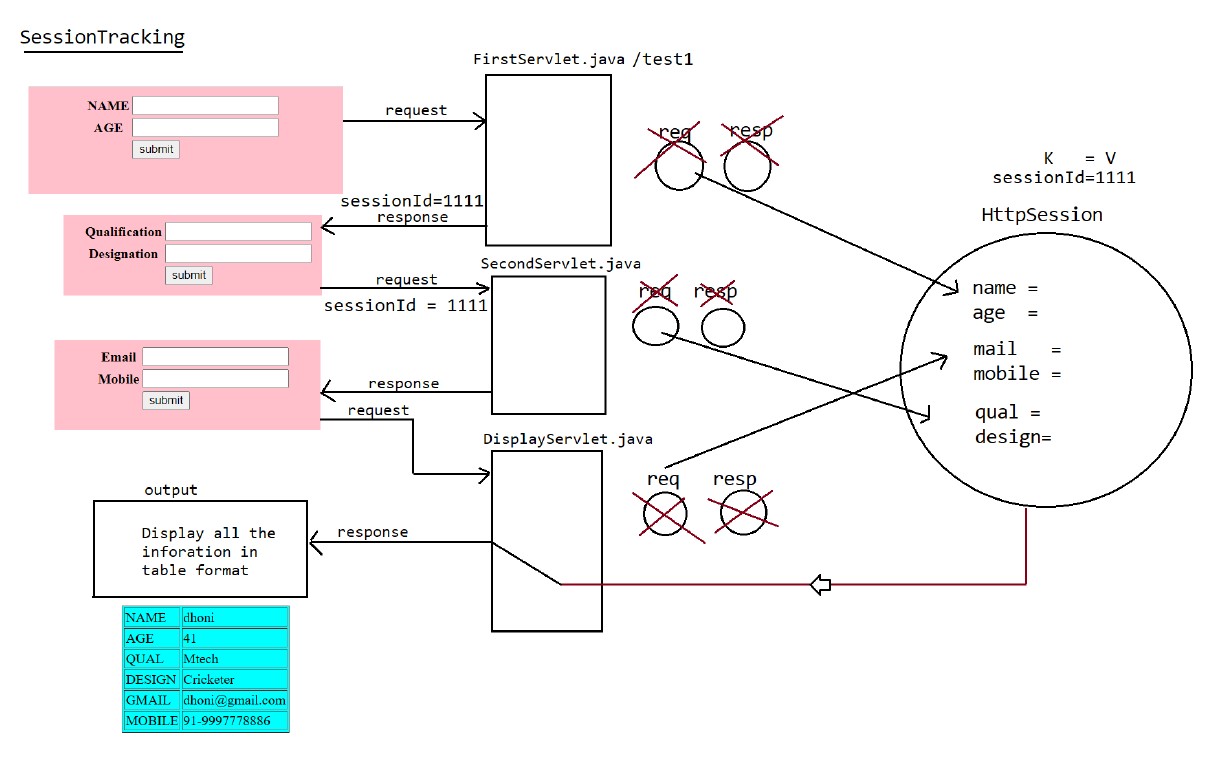
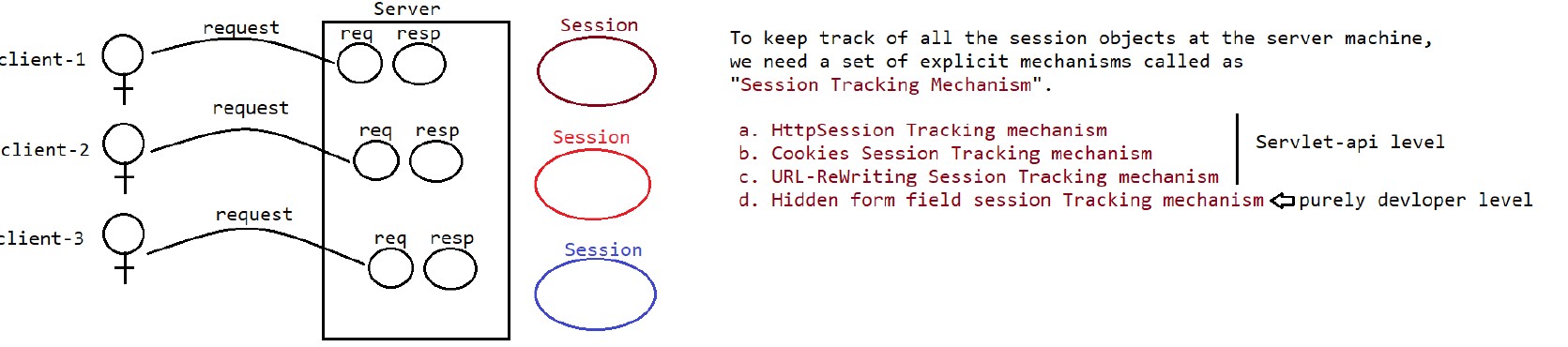
**What is Session?**

Session is a time duration, it will start from starting point of the client conversion with server and will terminate at the ending point of client conversion with the server.



Now suppose we have three client which is accessing our server, then how server will track the each session of the client.

The answer is simple each client has separate request and response object which is kept in separate session tracking mechanism. There are four session tracking mechanism through which we can track the session of the client.



Now suppose there are three html pages which request their corresponding servlet, our requirement is that we need to store the value of all html pages and then we need to present all the data of the three html pages in one pages in table format.

Now here one question arrises , all the servlet are stateless it mean it don’t store the value of the previos request if it directed to new request then what need to be done so that the all the parameters of html pages will be stored and presented in table form at last.

The answer is simple we need to use servlet session to store the value of all the parameter of the html page as shown in the figure and at last we need to represent in the form of table by taking all the value from the servlet session.

Creation of session for each user will be like:-

HttpSession session=request.getSession();

session.setAttribute("name", name);

session.setAttribute("age", age);

**getSession()->** The container will check whether any httpSession object existed for particular user or not. If any httpsession exists then the container will return the existed HttpSession object reference. If no httpSession exists is existed for particular user then container will create a new httpSession object and return the reference.

**getSession(Boolean)->** The container will check whether any HttpSession object existed for particular user or not. If any httpSession exists then the container will return the existed HttpSession object reference. If no httpSession exists is existed for particular user then it would return null.

Q.How the HTTPSession container will know if the object of req and response object belongs to same client?

Ans:- In the HttpSession tracking mechanism it creates different session id for different client. So by matching the session we can only stores the parameter of it

Now let’s see the code of the first html:-

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Input page</title>**

**</head>**

**<body bgcolor="pink">**

**<center>**

**<form action="./test1" method="get">**

**<table>**

**<tr>**

**<th>Name</th>**

**<td><input type='text' name='name' placeholder='name'/></td>**

**</tr>**

**<tr>**

**<th>Age</th>**

**<td><input type='text' name='age' placeholder='age'/></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='submit'/></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

Now let’s see the code of corresponding servlet

**package in.ineuron.controller;**

**@WebServlet("/test1")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**String name = request.getParameter("name");**

**String age = request.getParameter("age");**

**//create the session object and store the request data in**

**//session object**

**HttpSession session=request.getSession();**

**session.setAttribute("name", name);**

**session.setAttribute("age", age);**

**//forward the response to the form2.html**

**RequestDispatcher rd = request.getRequestDispatcher("/form2.html");**

**rd.forward(request, response);**

**}**

**}**

Now let’s see the form2.html

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Email page</title>**

**</head>**

**<body bgcolor="pink">**

**<center>**

**<form action="./test2" method="get">**

**<table>**

**<tr>**

**<th>Qualification</th>**

**<td><input type='text' name='qalification' placeholder='Qualification'/></td>**

**</tr>**

**<tr>**

**<th>Designation</th>**

**<td><input type='text' name='design' placeholder='design'/></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='submit'/></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

and it’s corresponding servlet is given below:-

**package in.ineuron.controller;**

**import java.io.IOException;**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**String qualification = request.getParameter("qalification");**

**String designation = request.getParameter("design");**

**//create the session object and store the request data in**

**//session object**

**HttpSession session=request.getSession();**

**session.setAttribute("qualification", qualification);**

**session.setAttribute("designation", designation);**

**//forward the response to the form3.html**

**RequestDispatcher rd = request.getRequestDispatcher("/form3.html");**

**rd.forward(request, response);**

**}**

**}**

Now let’s see the form3.html

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Email page</title>**

**</head>**

**<body bgcolor="pink">**

**<center>**

**<form action="" method="get">**

**<table>**

**<tr>**

**<th>Email</th>**

**<td><input type='text' name='email' placeholder='email'/></td>**

**</tr>**

**<tr>**

**<th>Mobile</th>**

**<td><input type='text' name='Mobile' placeholder='Mobile'/></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='submit'/></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

Now let’s see the servlet page:-

**@WebServlet("/test3")**

**public class ThirdServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**String email = request.getParameter("email");**

**String mobile = request.getParameter("mobile");**

**//create the session object and store the request data in**

**//session object**

**HttpSession session=request.getSession();**

**session.setAttribute("email", email);**

**session.setAttribute("mobile", mobile);**

**String name=session.getAttribute("name").toString();**

**String age=session.getAttribute("age").toString();**

**String qual=session.getAttribute("qualification").toString();**

**String design=session.getAttribute("designation").toString();**

**String mail=session.getAttribute("email").toString();**

**String mob=session.getAttribute("mobile").toString();**

**//give response to the html**

**response.setContentType("text/html");**

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

**out.println("<html>");**

**out.println("<head>");**

**out.println("<title> Response page</title>");**

**out.println("</head>");**

**out.println("<body bgcolor='skyblue'>");**

**out.println("<center>");**

**out.println("<table border='1'>");**

**out.println("<form>");**

**out.println("<tr>");**

**out.println("<th>Name</th><td>"+name+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>AGE</th><td>"+age+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>QUAL</th><td>"+qual+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>DESIGN</th><td>"+design+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Email</th><td>"+mail+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>MOBILE</th><td>"+mob+"<td>");**

**out.println("</tr>");**

**out.println("</form>");**

**out.println("</table>");**

**out.println("</center>");**

**out.println("</body>");**

**out.println("</html>");**

**}**

**}**

**Disadvantages of HTTPSession**

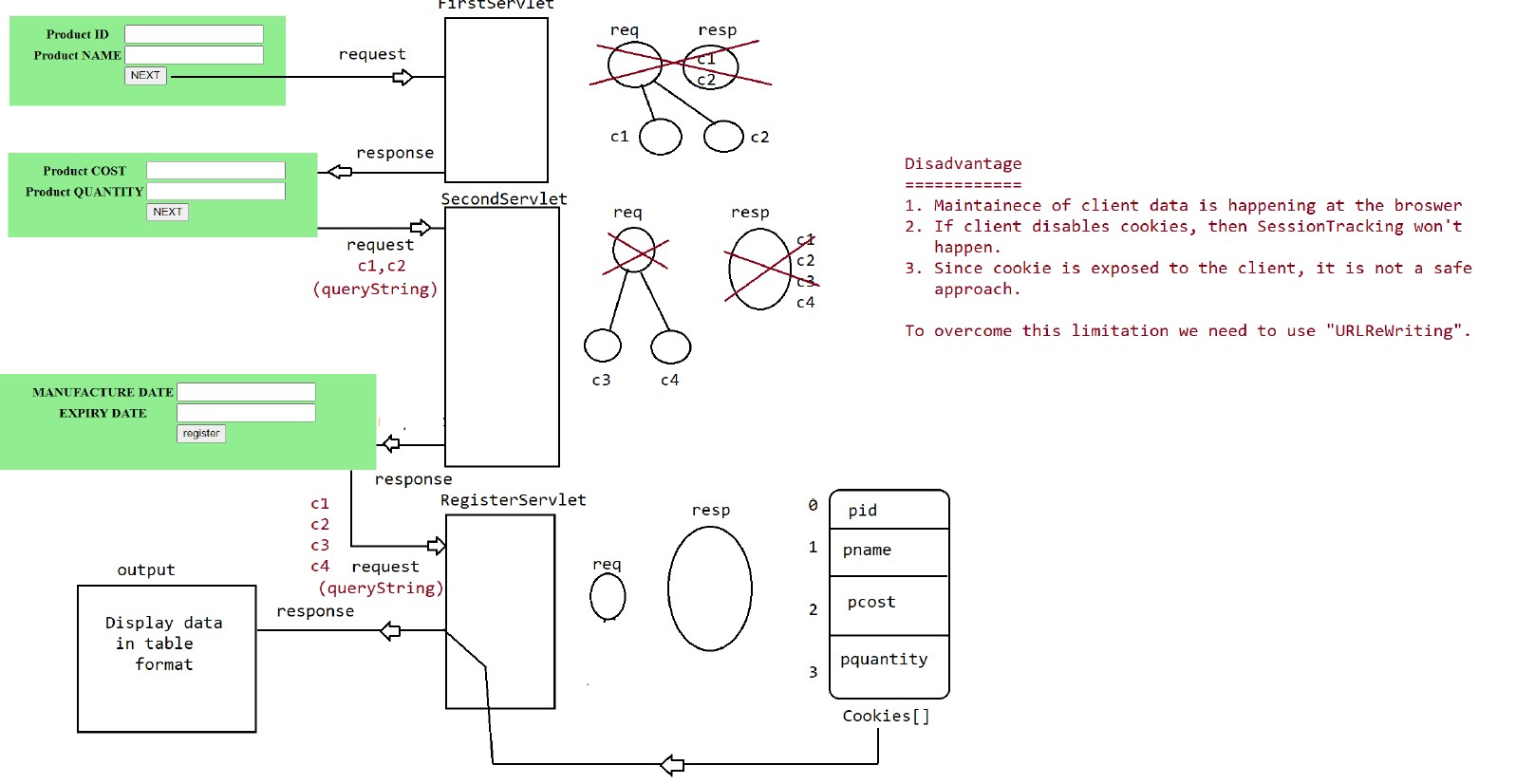
🡺More the no of users, more would be the session object.

🡺More the seesion objects, those objet will be in the server memory.

🡺More the session objects, maintainence woul be difficult at the server side

**To resolve this proble we use “cookieSessionTracking”** Mechanism**.**

**CookieSessionTracking**



🡺As we can see above, HTML page contains two parameter i.e. product id and Product Name which gives request to the FIrstServlet page which creates two cookies c1 and c2 for two parametes.

🡺From that FirstServlet page request will go to the Second HTML page from that page request will go to the second servlet page as second html page has two parameter it will create another two cookies i.e. c2 and c4 and earlies we have c1 and c2. So total we have c1,c2,c3 and c4.

🡺Now from secondservlet page gives the request to the third html page and this third html page gives request to the registedservlet which will create another two cookie for two parameter i.e. c5,c6 so in total there are 6 cookie c1,c2,c3,c4,c5,c6.

🡺From that we will display all the details in another page.

Let’s see how to create cookie for different parameter:-

**Cookie c1 = new Cookie("pid", productId);**

**Cookie c2 = new Cookie("pname", productName);**

To retrieve back the cookies from the request:-

**Cookie [] cookies =request.getCookies();**

**String pid =cookies[0].getValue();**

**String pname =cookies[1].getValue();**

**String pcost =cookies[2].getValue();**

**String pquantity =cookies[3].getValue();**

Now let’s see the first html page and the corresponding servlet page:-

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>First page</title>**

**</head>**

**<body bgcolor='lightgreen'>**

**<center>**

**<form action="./test1" method="get">**

**<table>**

**<tr>**

**<th>Product Id</th>**

**<td><input type='text' name='pid' placeholder='product Id' /></td>**

**</tr>**

**<tr>**

**<th>Product Name</th>**

**<td><input type='text' name='pname' placeholder='product Name' /></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='Next' /></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

Now let’s see the servlet page where we need to create the cookie:-

**@WebServlet("/test1")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**//collect request Parameter**

**String productId =request.getParameter("pid");**

**String productName=request.getParameter("pname");**

**//create a cookie as per the requirement(inputs)**

**Cookie c1 = new Cookie("pid", productId);**

**Cookie c2 = new Cookie("pname", productName);**

**//send the cookie object through response**

**response.addCookie(c1);**

**response.addCookie(c2);**

**RequestDispatcher rd =request.getRequestDispatcher("/form2.html");**

**rd.forward(request, response);**

**}**

**}**

Now let’s see the second html page i.e.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Second Page</title>**

**</head>**

**<body bgcolor='lightgreen'>**

**<center>**

**<form action='./test2' method='get'>**

**<table>**

**<tr>**

**<th>Product Quantity</th>**

**<td><input type='text' name='pquantity'**

**placeholder='product Quantity' /></td>**

**</tr>**

**<tr>**

**<th>Product cost</th>**

**<td><input type='text' name='pcost' placeholder='product Cost' /></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='Next' /></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

Now let’s see the second servlet:-

**@WebServlet("/test2")**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**//collect request Parameter**

**String productQuantity =request.getParameter("pquantity");**

**String productCost=request.getParameter("pcost");**

**//create a cookie as per the requirement(inputs)**

**Cookie c1 = new Cookie("pquantity", productQuantity);**

**Cookie c2 = new Cookie("pcost", productCost);**

**//send the cookie object through response**

**response.addCookie(c1);**

**response.addCookie(c2);**

**RequestDispatcher rd =request.getRequestDispatcher("/form3.html");**

**rd.forward(request, response);**

**}**

**}**

Now let’s see the third html page:-

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Second Page</title>**

**</head>**

**<body bgcolor='lightgreen'>**

**<center>**

**<form action='./test3' method='get'>**

**<table>**

**<tr>**

**<th>Manufacture Date</th>**

**<td><input type='text' name='man\_date'**

**placeholder='Manufacture Date' /></td>**

**</tr>**

**<tr>**

**<th>Expiry date</th>**

**<td><input type='text' name='Exp\_date' placeholder='Exp\_Date' /></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='register' /></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

And the flow of direction will go the **Register.servlet** let’s see the code of the register.servlet:-

**@WebServlet("/test3")**

**public class Register extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**//collect request Parameter**

**String manufactureDate =request.getParameter("man\_date");**

**String expiryDate=request.getParameter("Exp\_date");**

**//Collect the cookies from the request Object**

**Cookie [] cookies =request.getCookies();**

**String pid =cookies[0].getValue();**

**String pname =cookies[1].getValue();**

**String pcost =cookies[2].getValue();**

**String pquantity =cookies[3].getValue();**

**//sending the response throught html page:-**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table bgcolor='lightGreen' border=1>");**

**out.println("<form>");**

**out.println("<tr>");**

**out.println("<th>Product id</th><td>"+pid+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Product Name</th><td>"+pname+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Product Cost</th><td>"+pcost+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Product Quantity</th><td>"+pquantity+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Manufacture Date</th><td>"+manufactureDate+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Expire Date</th><td>"+expiryDate+"<td>");**

**out.println("</tr>");**

**out.println("</form>");**

**out.println("</center>");**

**out.println("</table>");**

**out.close();**

**}**

**}**

Disadvantages of cookies

🡺Maintainece of client data is happening at the browser

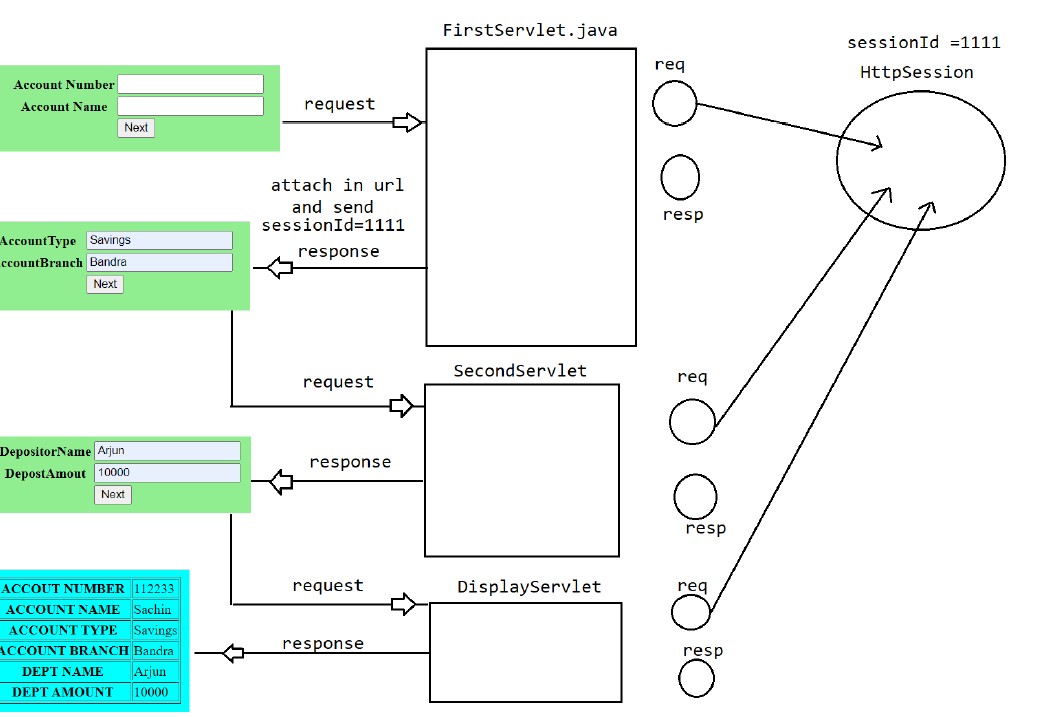
🡺If Client disable cookies, then SessionTracking wouldnot happen.

🡺Since cookie is exposed to the client, it is not a safe approach.

Q Why do we need URLRewriting over session tracking and cookieReTracking?

Ans :- We need URLRewriting over session tracking and cookieReTracking because incase of session tracking and CookieRetracking sessionId is generated implicitely in the browser if user disable the cookie in their browser it leads it create new session id for every session and it will create problem in it.

But in case of URLRewriting we creates session id explicitly in the browser and then pass to the server to process it.



**To overcome this limitation we need to use “URLReWriting”**

We pass session id in the action part of form or table from where the flow will go to the next page:-

Let’s see the code of it:-

Html Code

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Login page</title>**

**</head>**

**<body bgcolor='lightgreen'>**

**<center>**

**<form action='./test1'>**

**<table>**

**<tr>**

**<th>Account Number</th>**

**<td><input type='text' name='accNo'**

**placeholder='Account Number' /></td>**

**</tr>**

**<tr>**

**<th>Account Name</th>**

**<td><input type='**

**text' name='accName'**

**placeholder='Account Name' /></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='Next' /></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

**First Servlet Code**

**@WebServlet("/test1")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**String accNO=request.getParameter("accNo");**

**String accName=request.getParameter("accName");**

**HttpSession session =request.getSession();**

**session.setAttribute("accNO", accNO);**

**session.setAttribute("accName", accName);**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table border =1>");**

**//URLRewriting**

**out.println("<form method ='get' action='"+response.encodeURL("./test2")+"'>");**

**out.println("<tr><th>AccountType</th><td><input type ='text' name ='accType'/></td></tr>");**

**out.println("<tr><th>AccountBranch</th><td><input type ='text' name ='accBranch'/></td></tr>");**

**out.println("<tr><td><input type='submit' value='Next' /></td></tr>");**

**out.println("</table>");**

**out.println("</center>");**

**out.close();**

**}**

**}**

**SecondServlet**

**@WebServlet("/test2")**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**String accType=request.getParameter("accType");**

**String accBranch=request.getParameter("accBranch");**

**HttpSession session =request.getSession();**

**session.setAttribute("accType", accType);**

**session.setAttribute("accBranch", accBranch);**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table border =1>");**

**out.println("<form method ='get' action='"+response.encodeURL("./test3")+"'>");**

**out.println("<tr><th>DepositoryName</th><td><input type ='text' name ='depName'/></td></tr>");**

**out.println("<tr><th>DepostAmount</th><td><input type ='text' name ='depAmount'/></td></tr>");**

**out.println("<tr><td><input type='submit' value='Next' /></td></tr>");**

**out.println("</table>");**

**out.println("</center>");**

**out.close();**

**}**

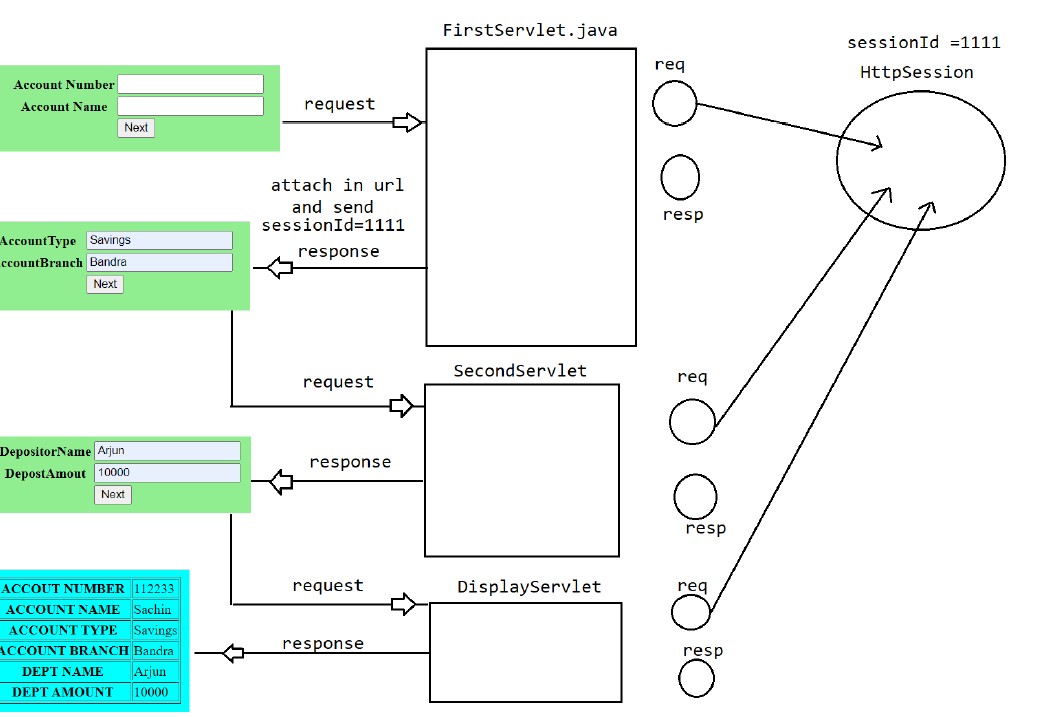
**}**

**Third Servlet**

**@WebServlet("/test3")**

**public class ThirdServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**



**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**String depName=request.getParameter("depName");**

**String depAmount=request.getParameter("depAmount");**

**//get the data from session object**

**HttpSession session =request.getSession();**

**Object accNo=session.getAttribute("accNO");**

**Object accName=session.getAttribute("accName");**

**Object accType=session.getAttribute("accType");**

**Object accBranch=session.getAttribute("accBranch");**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table bgcolor='lightGreen' border=1>");**

**out.println("<form>");**

**out.println("<tr>");**

**out.println("<th>Department Name</th><td>"+depName+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Department Amount</th><td>"+depAmount+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Account Number</th><td>"+accNo+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Account Name</th><td>"+accName+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Account Type</th><td>"+accType+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Account Branch</th><td>"+accBranch+"<td>");**

**out.println("</tr>");**

**out.println("</form>");**

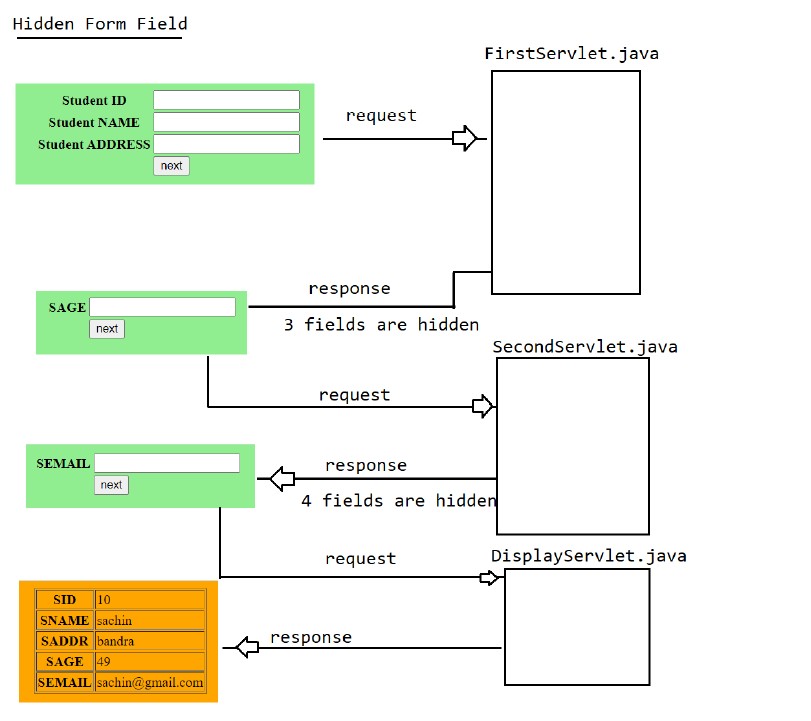
**out.println("</center>");**

**out.close();**

**}**

**}**

**Hidden Form Field**



We need to create hidden column in order to pass the value like:-

**<tr>**

**<th>**

**Student id**

**</th>**

**<td>**

**<input type =’hidden’ name=’sid’ value=’ ”+sid+” ‘/></td></tr>”);**

**<td>**

**</tr>**

**Let’s see the index html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Login Page</title>**

**</head>**

**<body bgcolor='lightgreen'>**

**<center>**

**<form action=''>**

**<table>**

**<tr>**

**<th>Student id</th>**

**<td><input type='text' name='sid'**

**placeholder='Student id' /></td>**

**</tr>**

**<tr>**

**<th>Student Name</th>**

**<td><input type='text' name='sname'**

**placeholder='Student Name' /></td>**

**</tr>**

**<tr>**

**<th>Student Address</th>**

**<td><input type='text' name='saddress'**

**placeholder='Student Address' /></td>**

**</tr>**

**<tr>**

**<td><input type='submit' value='Next' /></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

**Let’s see the first Servlet**

**@WebServlet("/test1")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**//collect the data from the request object**

**String sid =request.getParameter("sid");**

**String sname =request.getParameter("sname");**

**String saddr =request.getParameter("saddr");**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table bgcolor='lightGreen' border=1>");**

**out.println("<form action ='./test2'>");**

**out.println("<tr>");**

**out.println("<th>Student Age</th><td><input type ='text' name ='sage' placeholder='Student Age'/><td>");**

**out.println("<tr><th>Student ID</th><input type ='hidden' name ='sid' value='"+sid+"'><td></tr>");**

**out.println("<tr><th>Student Name</th><input type ='hidden' name ='sname' value='"+sname+"'><td></tr>");**

**out.println("<tr><th>Student Address</th><input type ='hidden' name ='saddr' value='"+saddr+"'><td></tr>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<td><input type='submit' value='Next'/><td>");**

**out.println("</tr>");**

**out.println("</form>");**

**out.println("</center>");**

**out.close();**

**}**

**Let’s see the Scond Servlet**

**@WebServlet("/test2")**

**public class SecondServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**//collect the data from the request object**

**String sid =request.getParameter("sid");**

**String sname =request.getParameter("sname");**

**String saddr =request.getParameter("saddr");**

**String sage =request.getParameter("sage");**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table bgcolor='lightGreen' border=1>");**

**out.println("<form action ='./test3'>");**

**out.println("<tr>");**

**out.println("<th>Student Age</th><td><input type ='hidden' name ='sage'value='"+sage+"'/><td>");**

**out.println("<tr><th>Student ID</th><input type ='hidden' name ='sid' value='"+sid+"'/><td></tr>");**

**out.println("<tr><th>Student Name</th><input type ='hidden' name ='sname' value='"+sname+"'/><td></tr>");**

**out.println("<tr><th>Student Address</th><input type ='hidden' name ='saddr' value='"+saddr+"'/><td></tr>");**

**out.println("<th>Student Email</th><td><input type ='text' name ='semail' placeholder='Student Email'/><td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<td><input type='submit' value='Next'/><td>");**

**out.println("</tr>");**

**out.println("</form>");**

**out.println("</center>");**

**out.close();**

**}**

**}**

**Let’s see the third servlet**

**@WebServlet("/test3")**

**public class ThirdServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**// TODO Auto-generated method stub**

**//collect the data from the request object**

**String sid =request.getParameter("sid");**

**String sname =request.getParameter("sname");**

**String saddr =request.getParameter("saddr");**

**String sage =request.getParameter("sage");**

**String semail =request.getParameter("semail");**

**response.setContentType("text/html");**

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

**out.println("<center>");**

**out.println("<table bgcolor='lightGreen' border=1>");**

**out.println("<form>");**

**out.println("<tr>");**

**out.println("<th>Student Name</th><td>"+sname+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Student id</th><td>"+sid+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Student address</th><td>"+saddr+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Student Age</th><td>"+sage+"<td>");**

**out.println("</tr>");**

**out.println("<tr>");**

**out.println("<th>Student Email</th><td>"+semail+"<td>");**

**out.println("</tr>");**

**out.println("</form>");**

**out.println("</center>");**

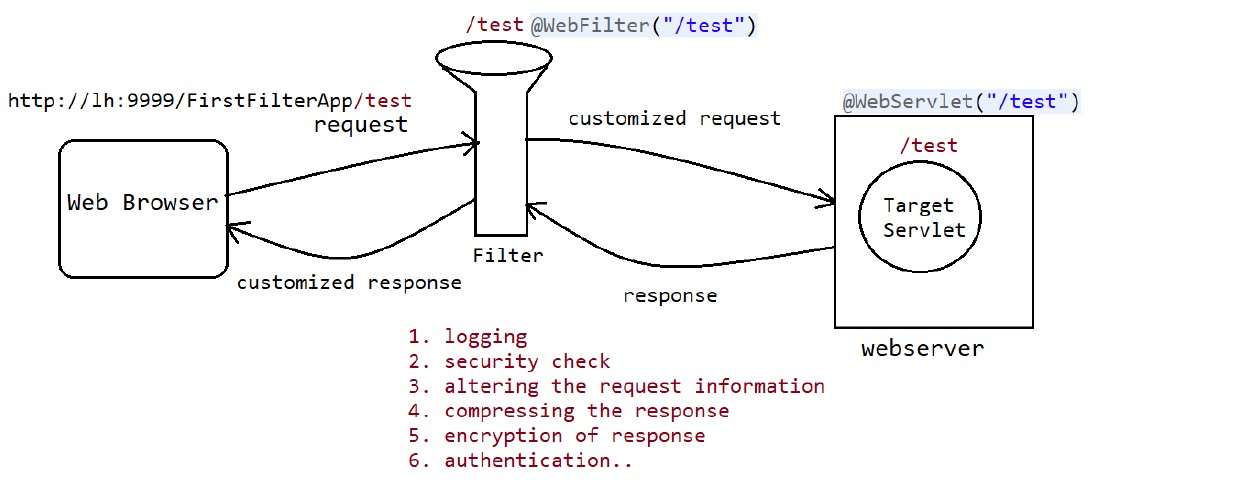
**out.close();**

**}**

**}**

**Filter and Listener in MVC**

**Filter**



Filter is used for preprocessing the request and post processing of request before they reach the target resource of the webapplication.

**Note:-**

Whenever we sending the request to target Servlet , container will check whether any filter is configured for this servlet or not.

If fileter is configured then container will forward the request to filter instead of servlet.

**Void doFilter(req,resp,chain) throws SE,IOE**

After executing the above logic, filter forwards the request to targetServlet.

After processing the request by TargetServlet the response will be forwarded to the filter instead of browser and after executing the filter logic , filter forwards the total response to the browser.

@webfilter(“ “) is used incase of filter

@webServlet(“ “) is used in case of servlet

It is used for:-

🡺Logging

🡺Security check

🡺Altering the request information

🡺Compressing the response

🡺Encryption of response

🡺Authentication

Filter is created by extending HttpFilter and implementing Filter and it contains three methods namely:-

**🡺init method 🡪**Filter object is initialised

**🡺dofillter🡪**Filter processing and filter post processing logic is codded here

**🡺destroy🡪**It will deinstantiated the filter object.

**The URL Pattern for both filter and controller should be same**

Let’ see the code to know how to use filter:-

First we will see the filter:-

**package in.ineuron.filter;**

**@WebFilter("/test1")**

**public class DemoFilter extends HttpFilter implements Filter {**

**static {**

**System.out.println("Filter is loading");**

**}**

**public DemoFilter() {**

**System.out.println("Filter object is instantiated");**

**}**

**public void init(FilterConfig fConfig) throws ServletException {**

**System.out.println("Filter Initialization");**

**}**

**public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)**

**throws IOException, ServletException {**

**//Filter preprocessing logic is coded**

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

**out.println("<h1>This line is added by demo filter before processing the request....</h1>");**

**//pass the request along the filter chain**

**chain.doFilter(request, response);**

**//filter post processing is coded here.**

**out.println("<h1>This line is added by demo filter after post processing the request....</h1>");**

**out.close();**

**}**

**public void destroy() {**

**System.out.println("filter object is deinstantiated");**

**}**

**}**

🡺 As we can see in the **doFilter** method , there is first preprocessing logic which contain the code which is not processed.

🡺If the filter is configured properly in the **dofilter method** it will forward the request to the servlet by using method

**doFilter(req,resp,chain)**

Now let’ s see the code of servlet:-

**@WebServlet("/test1")**

**public class TergetServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**static {**

**System.out.println("servlet loading");**

**}**

**public void TargetServlet() {**

**System.out.println("servlet instntiation.....");**

**}**

**@Override**

**public void init() throws ServletException {**

**System.out.println("servlet initialisation.....");**

**}**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException**

**{**

**System.out.println("Request Processing Phase.....");**

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

**out.println("<h1>This is the response from the Target Servlet</h1>");**

**}**

**@Override**

**public void destroy() {**

**System.out.println("Servlet De-instntiation");**

**}**

**}**

🡺After processing the the request by Target servlet, the respose will be forwarded to the filter instead of browser and after executing the filter logic, filter forwards the total responst to the browser.

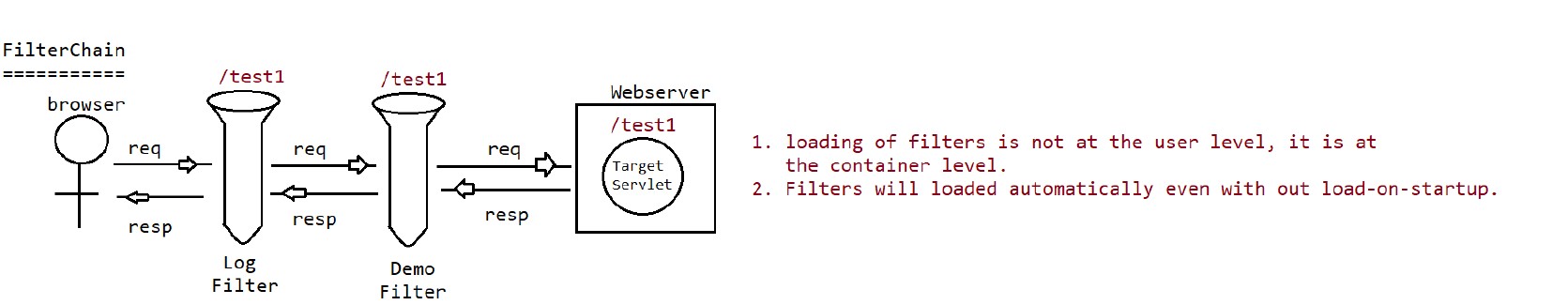
Output:-

**This line is added by demo filter before processing the request....**

**This is the response from the Target Servlet**

**This line is added by demo filter after post processing the request....**

**Filter Chaining**



A filter is an object that is used throughout the pre-and post-processing stages of a request. Conversion, logging, compression, encryption and decryption, input validation, and other filtering operations are commonly performed using it.

### **Servlet Filter Chain**

We will learn how to correlate a chain of filters with a web resource in this lesson. A Servlet, JSP, or static HTML page might be used as the web resource. We may conduct numerous operations within a web application using the filter function, such as session validation, user authentication and verification, blocking access to a certain online resource, and so on. By implementing the Filter interface, we may build several Filter classes. When a user makes a request to a web resource after associating several Filter classes with it, the associated Filter classes are run before and after the web resource requested by the user is processed, thereby giving a filter to the user request and response. Multiple filters associated with a web resource are performed in the order provided in the deployment descriptor file’s filter-mapping> tag (web.xml).

**Methods use for filter chain interface:**

To develop a filter class, we must implement the three **javax.servlet.Filter** Interface for filtering.

1. **void doInit(FilterConfig config) –** The Filter is initialized with this method.
2. **void doFilter(ServletRequest request, ServletResponse, response, FilterChain chain) –** When a client requests a web resource, such as a Servlet or a JSP page, the web container calls this function.
3. **void destroy() –** The Filter object is destroyed using this procedure.

**Let’s see the example first:**-

**LogFilter..html**

**@WebFilter("/test1")**

**public class LogFilter extends HttpFilter implements Filter {**

**private FilterConfig config;**

**static {**

**System.out.println("LogFilter Loading.....");**

**}**

**public LogFilter() {**

**System.out.println("LogFiler instantiation.....");**

**}**

**public void init(FilterConfig config) throws ServletException {**

**System.out.println("LogFilter instialisation....");**

**this.config = config;**

**}**

**public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)**

**throws IOException, ServletException {**

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

**out.println("<h1>This line is added by the logfilter before processing the Request</h1>");**

**ServletContext context = config.getServletContext();**

**HttpServletRequest req = (HttpServletRequest) request;**

**context.log("A request is coming from ::" + req.getRemoteHost() + "for URL ::" + req.getRequestURI()**

**+ "at date :: " + new java.util.Date());**

**// pass the request along the filter chain**

**chain.doFilter(request, response);**

**out.println("<h1>This line is added by log filter after processing the request.....</h1>");**

**}**

**public void destroy() {**

**config = null;**

**System.out.println("LogFilter Destroy");}}**

Let’s see the second filter

**@WebFilter("/test1")**

**public class DemoFilter extends HttpFilter implements Filter {**

**static {**

**System.out.println("DemoFilter Loading.....");**

**}**

**public DemoFilter() {**

**System.out.println("DemoFilter instantiation.....");**

**}**

**public void init(FilterConfig fConfig) throws ServletException {**

**System.out.println("DemoFilter instialisation....");**

**}**

**public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {**

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

**out.println("<h1>This line is added by Demofilter before processing the request</h1>");**

**// pass the request along the filter chain**

**chain.doFilter(request, response);**

**out.println("<h1>This line is added aby demofilter after processing</h1>");**

**}public void destroy() {**

**System.out.println("DemoFilter Destroy");**

**}}**

Let’s see the servlet:-

**@WebServlet("/test1")**

**public class TargetServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**static {**

**System.out.println("Target Servlet Loading....");**

**}**

**public TargetServlet() {**

**System.out.println("Target Servlet Instantiation......");**

**}**

**public void init() throws ServletException{**

**System.out.println("Target Servlet Instialisation......");**

**}**

**@Override**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

**System.out.println("Request Processing Phase");**

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

**out.println("<h1>This is the response from the target Servlet....</h1>");**

**}**

**@Overrid**

**public void destroy() {**

**System.out.println("Target Servlet Destroy() called......");**

**}}**

**Output:-**

# This line is added by Demofilter before processing the request

# This line is added by the logfilter before processing the Request

# This is the response from the target Servlet....

# This line is added by log filter after processing the request.....

# This line is added aby demofilter after processing

**Order of Execution of Filters in XMl:-**

🡺Identify all the filters which are configured in url-patterns and excute all these filters from top to bottoms

🡺Identify all the filters which are configured according to servlet-name and execute all these filters from top to bottom

**Orders of Execution of Filters w.r.t Annotations**

**🡺**Filters are executed based on the Alphabetical names of the Filter Names.

**Customized Filter:-**

Q:-Please create the web application which take the word and check if the word is java,spring and jee then it will return Ineuron else it will return the same word.

Ans:- First we will create the html page which take input as word and then we will create the filter which take the input from the filter and pass it to the customized class which acts as request object. We create that class by extending the HttpServletRequestWrapper. In the filter class we create the object of customized class and pass the request parameter in it .

After passing the request parameter, the request will go to the customized class where we have override the get parameter method to full fill the condition.

And then request will go the servlet and from servlet response will get back to the filter and then comes back in the form of html.

Let’s first see the html page.

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Registration Page</title>**

**</head>**

**<body bgcolor="green" text="white">**

**<center>**

**<h1>INeuron Private Limited</h1>**

**</center>**

**<center>**

**<form action='./test' method='get'>**

**<table>**

**<tr>**

**<th>Enter any Word</th>**

**<td><input type='text' name='word' /></td>**

**</tr>**

**<tr>**

**<th></th>**

**<td><input type='submit' value='check' /></td>**

**</tr>**

**</table>**

**</form>**

**</center>**

**</body>**

**</html>**

Now let’s see the filter class:-

**@WebFilter("/test")**

**public class BadWordFIlter extends HttpFilter implements Filter {**

**public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {**

**System.out.println("Preprocessing logic of filter");**

**// pass the request along the filter chain**

**CustomizedRequest req = new CustomizedRequest((HttpServletRequest)request);**

**chain.doFilter(req, response);**

**System.out.println("post processing logic of filter");**

**}**

**}**

**Now let’s see the customized class which is extending javax.servlet.http.HttpServletRequestWrapper**

**public class CustomizedRequest extends HttpServletRequestWrapper {**

**public CustomizedRequest(HttpServletRequest request) {**

**super(request);**

**}**

**@Override**

**public String getParameter(String name) {**

**// TODO Auto-generated method stub**

**String ineuron = "ineuron";**

**String word = super.getParameter(name);**

**if (word.equalsIgnoreCase("spring") || word.equalsIgnoreCase("java") || word.equalsIgnoreCase("jee")) {**

**return ineuron;**

**} else**

**return word;**

**}**

**}**

**Now from here we learn one core java concept i.e. if the super class has parameterized constructor, the subclass should have parameterized subclass.**

Now atLast we see the targetservlet:-

**@WebServlet("/test")**

**public class TargetServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**@Override**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

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

**String word =request.getParameter("word");**

**out.println("<h1>yout type word is...."+word+"</h1>");**

**}**

**}**

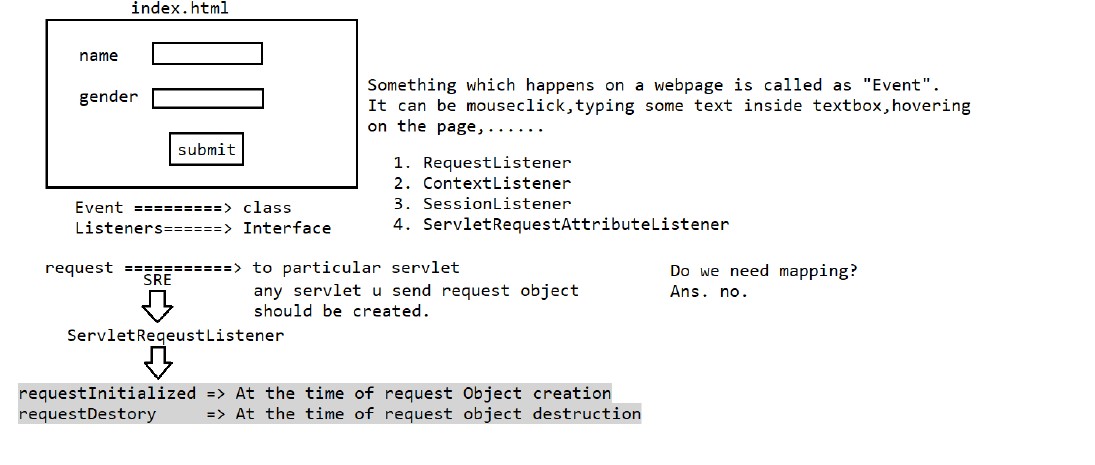
**Listeners**

Before understanding to the listeners we need to understand what is **Event.**

**Event** is something which happens in the webpages , it can be mouse click,typing some text inside textbox,hovering on the page…

Whenever event takes place in the webpage , simultenously one request object is created. Now the question arises how the request object is created automatically whenever there is any event?

There is **Listener**



which is kind of the interface, which look after the event.

Listeners are created by extending **ServletRequestListener:-**

It has many lifecly methods:-

Like

**RequestInitialised🡪** This method automatically get executed whenever there is any event in the servlet.

**RequestDestroyed🡪** This will destroy the listeners

Let’s see the servlet:-

**@WebServlet("/test")**

**public class FirstServlet extends HttpServlet {**

**private static final long serialVersionUID = 1L;**

**@Override**

**public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {**

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

**out.println("<h1>The no. of hit happen in the list" +RequestDemoListener.count+"</h1>");**

**}**

**}**

Let’s see the Listener:-

**@WebListener**

**public class RequestDemoListener implements ServletRequestListener {**

**public static int count=0;**

**public void requestDestroyed(ServletRequestEvent sre) {**

**// TODO Auto-generated method stub**

**System.out.println("request object destroyed :: "+new java.util.Date());**

**}**

**public void requestInitialized(ServletRequestEvent sre) {**

**// TODO Auto-generated method stub**

**count++;**

**System.out.println("Request object got created at ::"+new java.util.Date());**

**System.out.println("no of hit for request "+ count);**

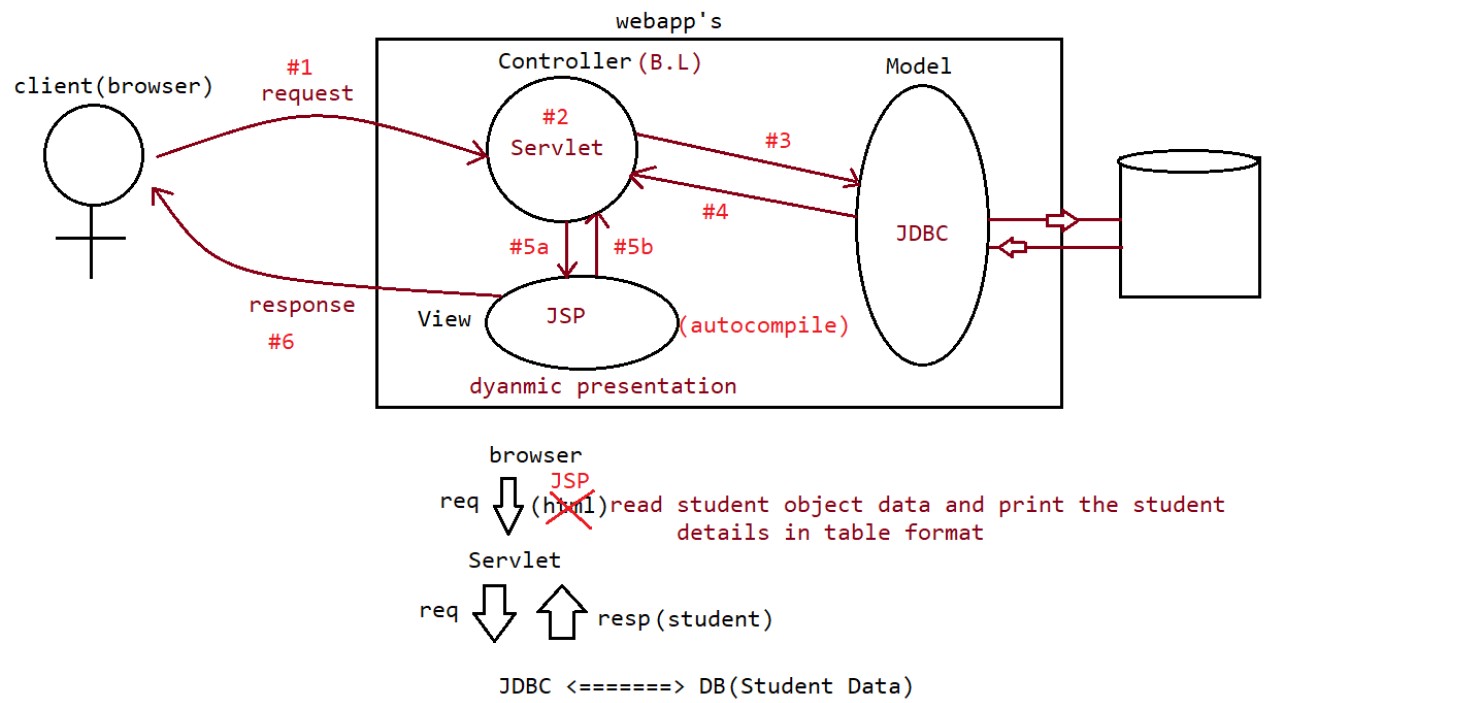
**}**

**}**

We donot need to map listener as it automatically get executed whenever there is any event. We only have to configure which kind of listener we want to use servlet, jsp or something else.

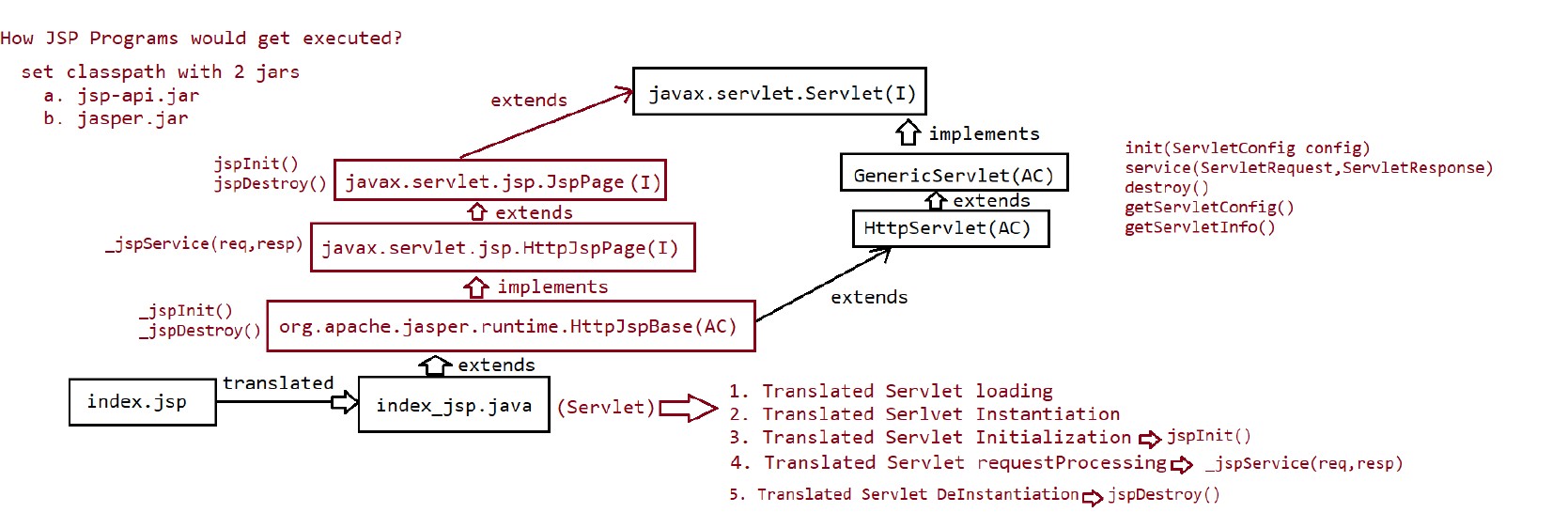
We need to map listener if and only if there is any kind of condition at which only listener should execute.

**Java Servlet Page**

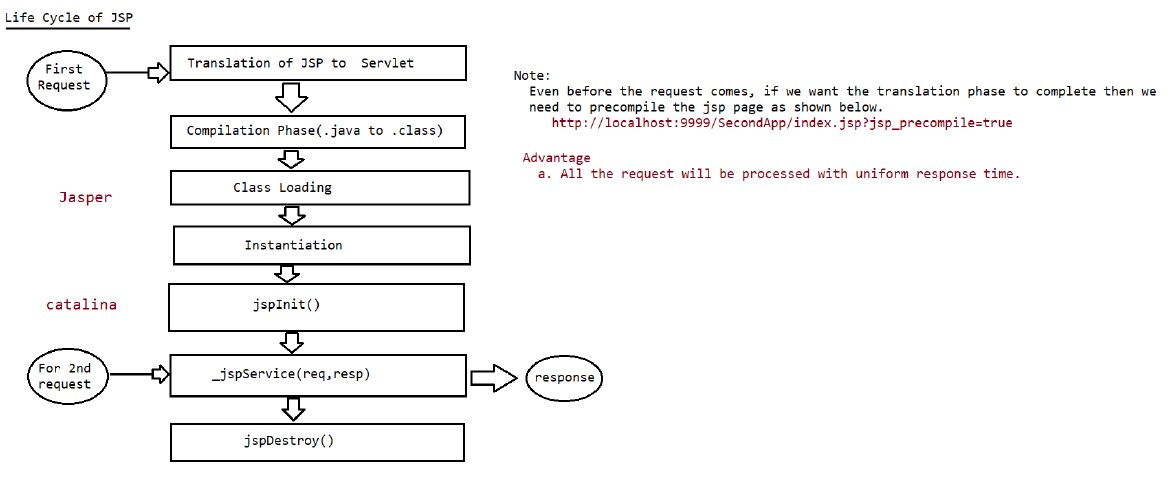


Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases

JSP is a dynamic web page which gets converted to java(Servlet) once container deploy the the jsp.



**Note:- UnderScore \_ sysmbol represents that this method is generating automatically by JSP engine.**



For the first request, all the process above will happen but during second request only processing will happen i.e. the control will go to the \_jspService directly.

Now before giving the request if we want to compile the jsp what we need to do is use ?after the jsp page and write down the **jsp\_precompile=true**

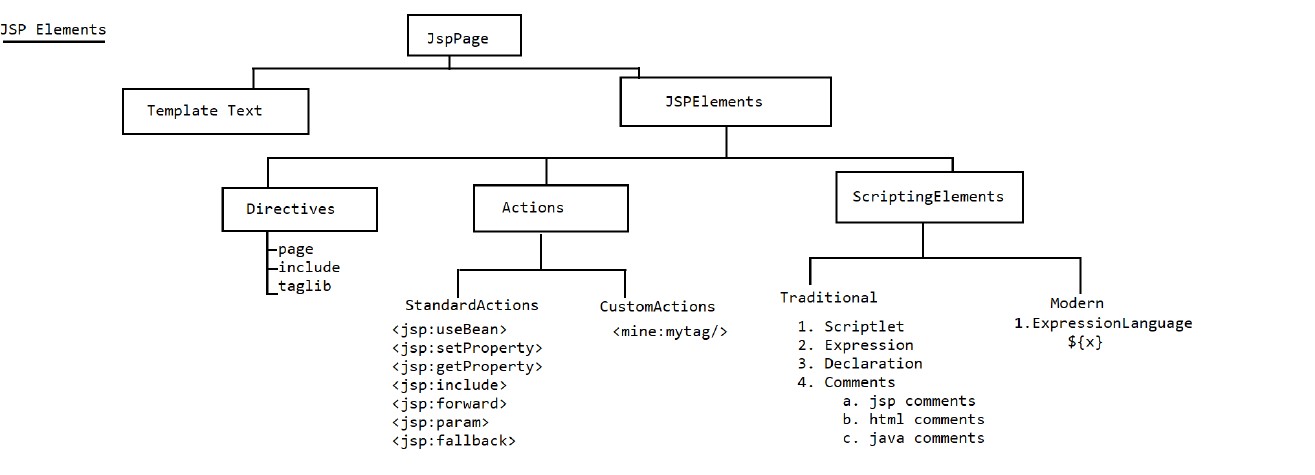
Let’s see the url for pre-compiling the jsp

[**http://localhost:9090/JSP1.0/index.jsp?jsp\_precompile=true**](http://localhost:9090/JSP1.0/index.jsp?jsp_precompile=true)

**Advantages**

All the request will be processed with uniform time

**Template Text**



🡺It contains plain text data and html tags

🡺For Template text no preprocessing is required and it will become arguments to Write() method in \_jspService(req,res) method.

Index.jsp

<h1> The server time is<%= new java.util.Date()%></h1>

Equivalent code for the above in \_jspService is

**Public final class index\_jsp extends ……{**

**\_jspService(request,response){**

**out.write(“the server time is “);**

**out.println(new java.util.date())**

**}**

**}**

**Write()🡺** It takes only charater data

**Println()🡺**it can take any type of argument

**Directives:-**

1. Page Directive:-

In the current jsp page if we want to define import statements,present jsp page characterstics then we need to go for page directive.

**JSP directives** are the elements of a JSP source code that guide the web container on how to translate the JSP page into its respective servlet.

**Syntax:-**

**<%@ page [attribute-list]%>**

**There are different attribute of jspdirective let’s see one by one:-**

a) **language=”java”**

This is the default value of language attribute

b)**Content Type=”text/html”**

This is the default value of contentType attribute.

As per the page requirement we can change the values too.

c)**import =’ ‘**

This is the only attribute which can be repeated in the following ways

Eg: <%@page import =’ ‘ import =’ ‘%>

[%@page import =’ java.io.\*,’java.util.Date’%](mailto:%25@page%20import%20='%20java.io.*,'java.util.Date'%25%20)

import ='java.util.Date,java.util.ArrayList'

<h1>

The server time is <%= new Date() %>

Storing Object inside <%=new ArrayList() %>

</h1>

The default values of import is

Java.lang.\*

Javax.servlet.\*

Javax.servlet.http.\*;

Javax.servlet.jsp.\*;

d)**extends**

The Default value of extends is HttpJspBase

e)**info**

The default value of this attribute is “jasper jsp2.3 Engine”

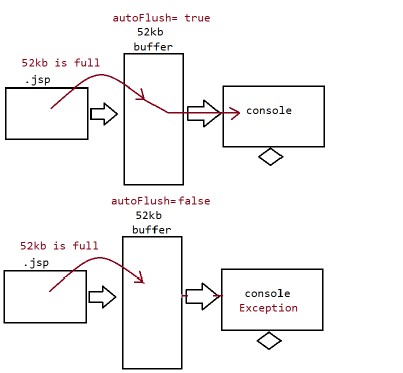
This value can be obtained inside the servlet by making a call to method called getServletInfo().

f)**buffer, autoFlush**

**Buffer🡺**JSP internally maintains a buffer to write the data to the console as the response. The default size of the buffer is 8kb.

**AutoFlush🡺** It is the Boolean attribute which is used to indicate the container whether to flush the response to cient automatically or not.

If autoFlush is true , then container will flush the complete reponse to the client from the buffer. If autoflush is false, then container will raise an exception when the Buffer is filled with response.



Java.io.IOException: Erro: JSP Buffer OverFlow.

**<body bgcolor='lightgreen' buffer ='52kb' autoflush='true'>**

**<center>**

**<b> <font size='7' color='red'><br />**

**<br /> <%**

**for(int i=0;i<=1000000;i++)**

**out.println("sachin");**

**%> </font>**

**</b>**

**</center>**

**</body>**

Suppose there is error in jsp page as we didn’t import some class in the jsp and we want that if error come in the jsp it will direct the control to the error.jsp where error is handle to do that we simple need to error page in page directive of the jsp like this:-

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8" errorPage="error.jsp"%>**

**<body bgcolor='lightgreen' buffer ='52kb' autoflush='true'>**

**<center>**

**<b> <font size='7' color='red'><br />**

**<br /> <%java.util.Date d=null;**

**out.println(d.toString());%> </font>**

**</b>**

**</center>**

**</body>**

Here error page is the attribute and we are adding the value as jsp page. Now suppose there is error in the the page and the flow of handed over to the jsp page.

**isErrorPage:**-This Attribute takes a default value as false, meaning the exception object would not be available inside this page to handle, to make the exception object available to jsp page we need to set the Boolean value to true.

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8" errorPage="error.jsp"%>**

**<body bgcolor='lightgreen' buffer ='52kb' autoflush='true'>**

**<center>**

**<b> <font size='7' color='red'><br /><br />**

**The exception occured is ::<%= exception %>**

**</font>**

**</b>**

**</center>**

**</body>**

**isELIgnored = “true” 🡺** Syntax wouldnot be processed rather it treats as Template Text.

**isELIgnored =”false” 🡺**Syntax will be processed and it print the value

Note:-default value is false,

<%@ page isELIgnored =”false”%>

<h1>

The username is :: ${param.user}<br/><br/>

The password is :: ${param.password}<br/><br/>

</h1>

**http:// localhost:9090/SecondApp/index.jsp?user=gaurav&password=Tendulkar**

**output**

The username is gaurav

The password is Tendulkar

Session Attribute:-

**<%@ page session =”false”%> false means in the current jsp page session object is not accessible. Default value is true.**

**<%**

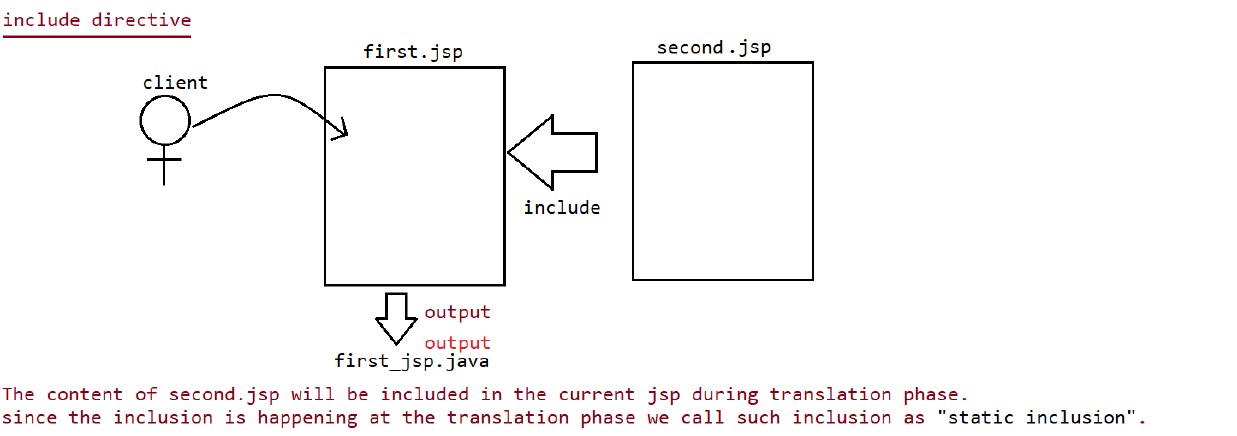
**session.setAttribute("Name","ineuron");**

**session.setAttribute("Java","NavinReddy");**

**%>**

**<h1> The name of the company is ::<%=session.getAttribute("Name") %></h1>**

**<h1> The Trainer name of the java is ::<%=session.getAttribute("Java") %></h1>**



**First JSP**

**<% @include file =”second.jsp”%>**

**<h1> This is First JSP<h1>**

**Second JSp**

**<h1> This is Second JSP<h1>**

Taglib Directive:-

It is customized tag which is created by the user.

<mine:iNeuron>

//body of the tag

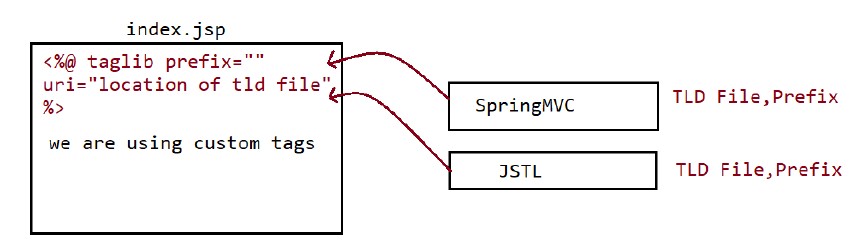
</mine:iNeuron>

When ever we do the customization, two files comes into picture:-

**1. TLD File(Tag library descriptor)**

**2.Prefix**

For few operations to be made easy for developers, the customized tags are already been coded by 3rd party vendor like SpringPivotal team, JSTL libraries by 3rd parties and so on…..



**Scripting Elements:-** If we want to write java code inside jsp at that time Scripting Element is requirement.

**Types of Scripting Elements**

A.Declarative tag

B.Scriptlet tag

C.Expression tag

D.Comments

a.html comments

b.jsp comments

c.java based comments

**Expression Tag**

This tag is used to put the value directly into the \_jspService() method in the **out.print(here)**

Syntax:-

<%=x %>🡺 \_jspservice(){

out.print(x);

}

**Scriptlet tag**

Syntax

<%

//write java code

%>

Like

**<h1> <%int i=0;**

**i++;**

**out.println(i);**

**%>**

**</h1>**

**Equivalent java Code**

🡺 \_jspservice(){

//java code scriplet code here

}

\_jspservice(){

**int i=0;**

**i++;**

**out.println(i);**

**}**

**Declarative Tag**

Syntax:

**<%!**

**//Any java declarations**

**%>**

The logic will be placed inside the servlet but ouside the \_jspservice(,,,,)

**Note:- Inside JSP, there are 9 implicit objects available and these objects are local to \_jspService() so these variables directly can’t be used inside declarative tag.**



**<h1>**

**<%!**

**int x=10;**

**static int y =20;**

**int[] a ={10,20,30};**

**public int m1(){**

**return x;**

**}**

**%>**

**The value of x is ::<%= m1() %><br>**

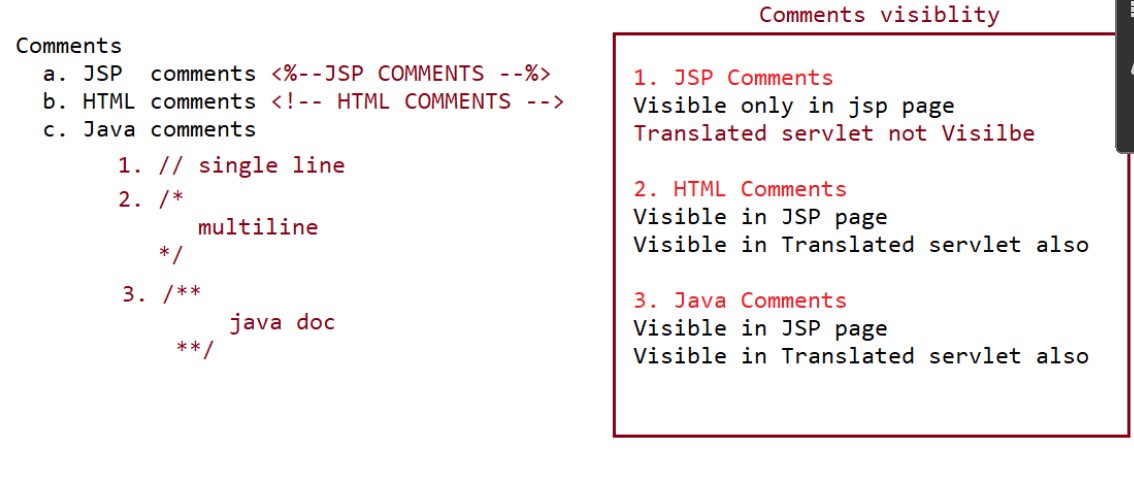
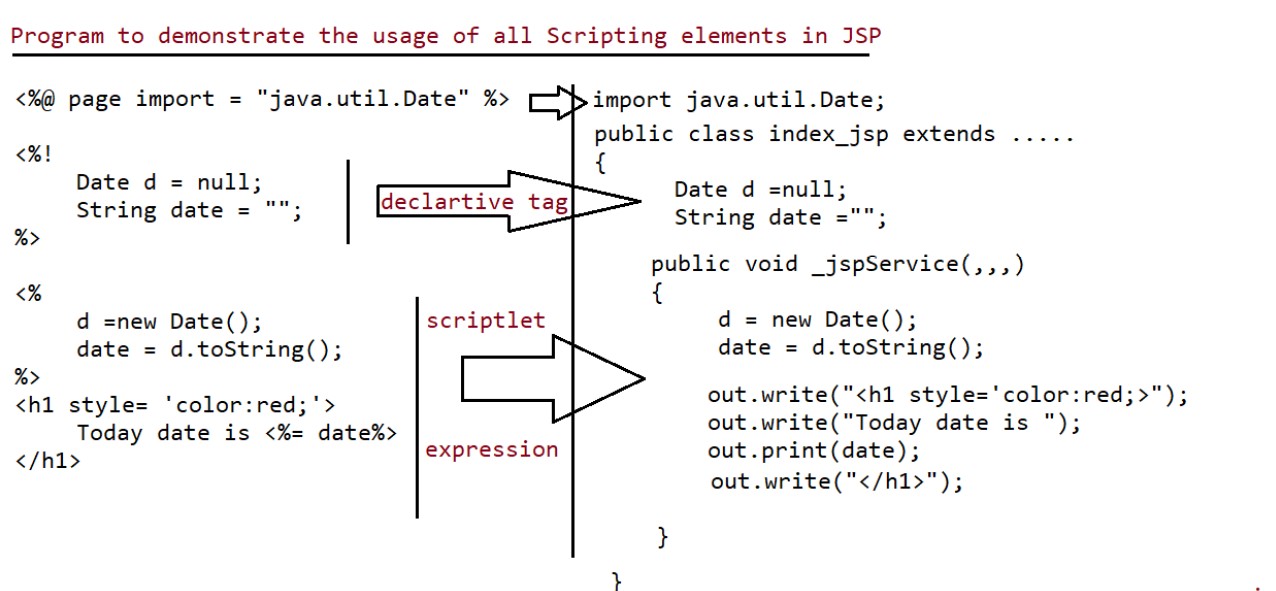
**The value of x is ::<%= y %><br>**

**</h1>**

**Output**

The value of x is ::10

The value of x is ::20



**Implicit Object**

**Response**

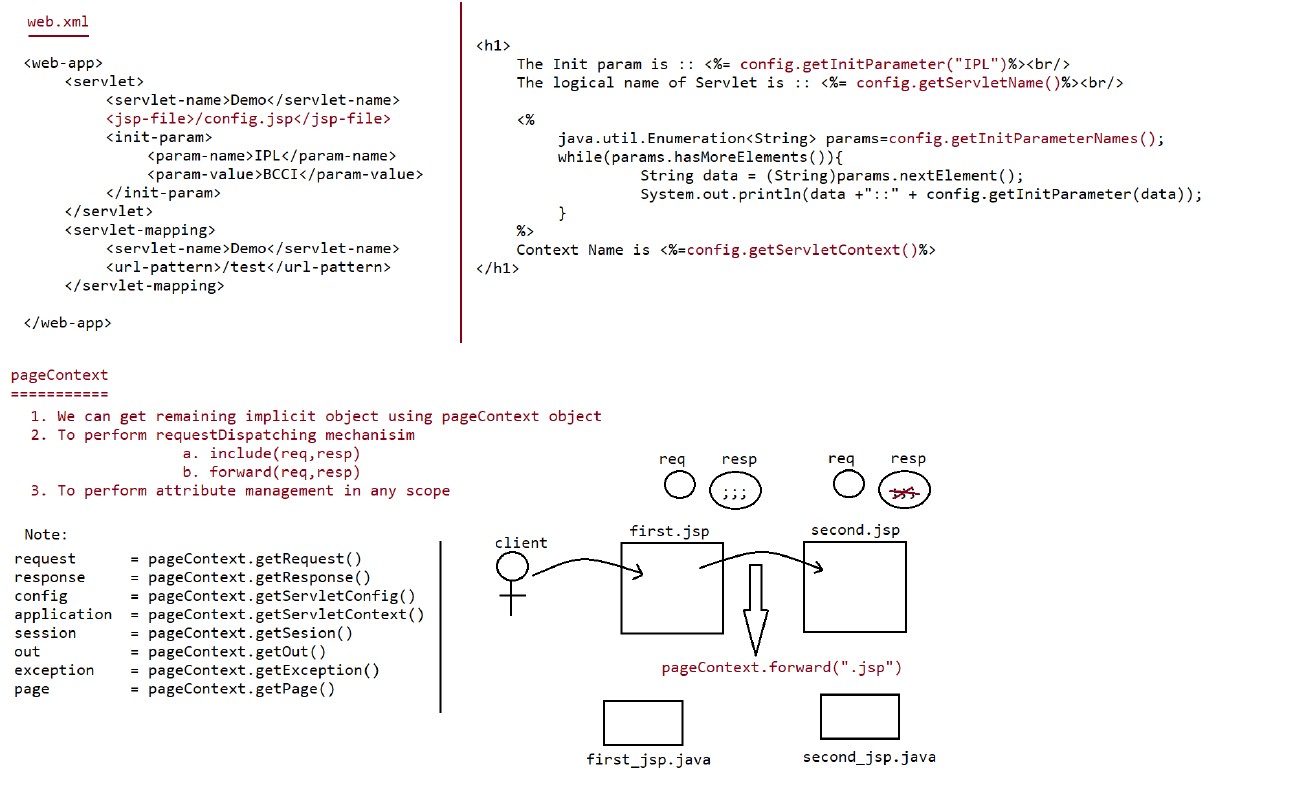
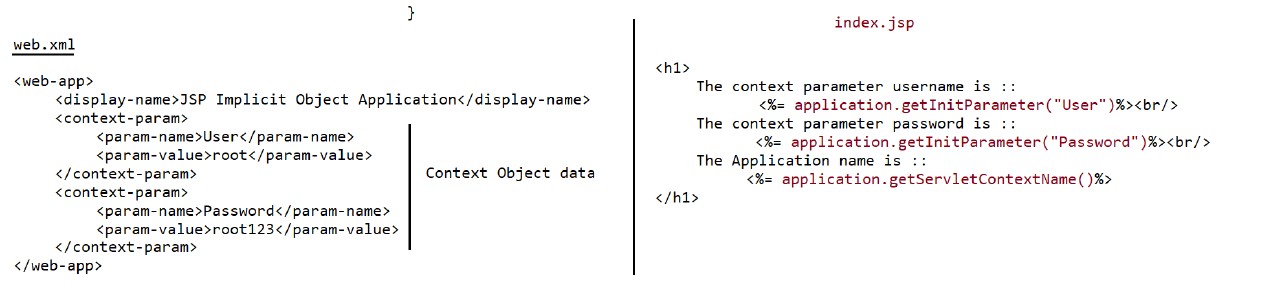
**<h1>Request method type is ::<%=request.getMethod() %></h1>**

**<h1>Username is :: <%=request.getParameter("user") %></h1>**

**<h1>Password is :: <%=request.getParameter("password")%></h1>**

**<h1>Password is :: <%=response.getContentType()%></h1>**

**Application implicit Function**



**PageContext Forward**

This is index page which is forwarding the request to the error.jsp

**<h1>This is the index jsp</h1>**

**<%**

**pageContext.forward("error.jsp");**

**%>**

**Second page code**

**<h1>This is the error page <h1>**

**PageContext include**

**<h1>This is the index jsp</h1>**

**<%**

**pageContext.include("error.jsp");**

**%>**

**<h1>Coming back control from the erro.jsp</h1>**

**Error page**

**<h1> This is the error page</h1>**

Scopes in JSP

1.Request Scope

2.Session Scope

3.Application Scope

4. Page Scope

**Request Scope**

In Servlet this scope in maintained by **ServletRequest Object**, In Jsp it is maintained by “request” implicit object. The information stored in request scope is available for all components which are processing that request.

Request Scope🡺It will start at time of request object creation(**before calling the service())**

It would end at the time of request object destruction(**after completing the service()**).

**Session Scope**

In servlet this is scope is maintained by “Httpservlet” object, In jsp it is maintained by “session” implicit object. The information stored in session scope is available for all components which are participating in session.

Session scope🡺It will start at athe time of session object creation.

It will end at the time of session expires(logout or invalidate()) or timeout mechanism.

**Application Scope**

In servlet this scope is mainted by “ServletContext” object, but in jsp it is maintained by “application” implicit object.

The information stored in application scope is available for all the components of the webapplication.

Application Scope🡺It will start at the time of ServletContext object (**at the time of server startup**)

It will end at the time of servletContext object destruction( **at the time of undeployment of server shut down**)

**Write a code using applicationscope to print hit count of the application**

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<%**

**Integer count = (Integer) application.getAttribute("hitcount");**

**if (count == null)**

**count = 1;**

**else**

**count++;**

**application.setAttribute("hitcount", count);**

**%>**

**<h1 style='color: red;'>**

**Hit count of the application is ::**

**<%=count%></h1>**

**</body>**

**</html>**

Q Write a code using application scope to count no. of user login to the application.

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<%**

**Integer count = (Integer) application.getAttribute("Usercount");**

**if(session.isNew()){**

**if (count == null)**

**count = 1;**

**else**

**count++;**

**}**

**application.setAttribute("Usercount", count);**

**%>**

**<h1 style='color: red;'>**

**Hit count of the application is ::**

**<%=count%></h1>**

**</body>**

**</html>**

Q Write a code to display the no of requests in current session?

Session-> hold the data uniquely w.r.t user, so keep it in session object.

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<%**

**Integer count = (Integer) session.getAttribute("sessionRequest");**

**if (session.isNew()) {**

**if (count == null)**

**count = 1;**

**else**

**count++;**

**}**

**session.setAttribute("sessionRequest", count);**

**%>**

**<h1 style='color: red;'>**

**user count of the application is ::**

**<%=count%></h1>**

**</body>**

**</html>**

**Note:-** In all the above program, initially the all the variables would not be available in the respective object, so null value will returned, based on the condition the variable would be created with the respective values and storedback in the respective objects.

**Page Scope**

This scope is not applicable for servlet, it is applicable for jsp.

In jsp this scope is managed by “pageContext” implicit object.

This information stored in page scope is available only in the present jsp page, and not available for other JSPs

**To retrieve the value form the PageContext Object W.R.T to the scope we need to use the following methods**

**a. pageContext.getAttribute(String name, int Scope);**

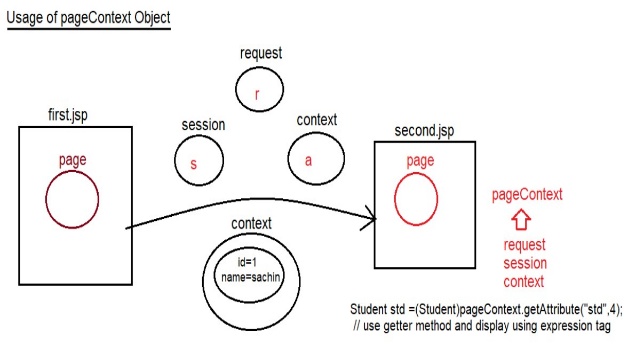
**The scope value of different scope are:-**

page Scope=1

request Scope =2

Session Scope=3

Application Scope =4



To retrieve the data in the page scope we need to feed respective scope in the pageContext .If we don’t specify the scope number while retrieving the value of the previous page we need to specify the scope no. Let’s see the code and understand:-

**First Jsp**

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Insert title here</title>**

**</head>**

**<body>**

**<%**

**pageContext.setAttribute("p", "page");**

**request.setAttribute("r", "request");**

**session.setAttribute("s", "session");**

**application.setAttribute("a", "application");**

**pageContext.forward("second.jsp");**

**%>**

**</body>**

**</html>**

**Second Jsp**

**<%@ page language="java" contentType="text/html; charset=UTF-8"**

**pageEncoding="UTF-8"%>**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="UTF-8">**

**<title>Insert title here</title>**

**</head>**

**<body>**

**Page Scope attribute::<%=pageContext.getAttribute("p",1)%><br>**

**Request Scope attribute ::<%=pageContext.getAttribute("r",2)%><br>**

**Session scope attribute ::<%=pageContext.getAttribute("s",3)%><br>**

**Application Scope Attribute ::<%=pageContext.getAttribute("a",4)%><br>**

**</body>**

**</html>**

**Output**

**Page Scope attribute :: null**

**Request Scope attribute :: request**

**Session Scope attribute :: session**

**Application Scope atribute :: application**

Difference between findAttribute(name) vs getAttribute(name)

getAttribute(name)=> By default it will check in page scope,until explicitly we tell through Scope.

getAttribute(name,scope)=> It will check in the respective scope, and if it is available it would return the value otherwise it would return null.

findAttribute(name)=> First it will check in page scope, followed by request scope, session scope, application scope. In any of this scope if the object is available it would return the value otherwise it would return null.

**<h1>Find Attribute ::<%=pageContext.findAttribute("a")%></h1>**

Usage of find Attribute

<%

pageContext.setAttribute("p","page");

request.setAttribute("r","request");

session.setAttribute("s","session");

application.setAttribute("a","application");

%>

<h1>Find Attribute ::<%=pageContext.findAttribute("a")%></h1>

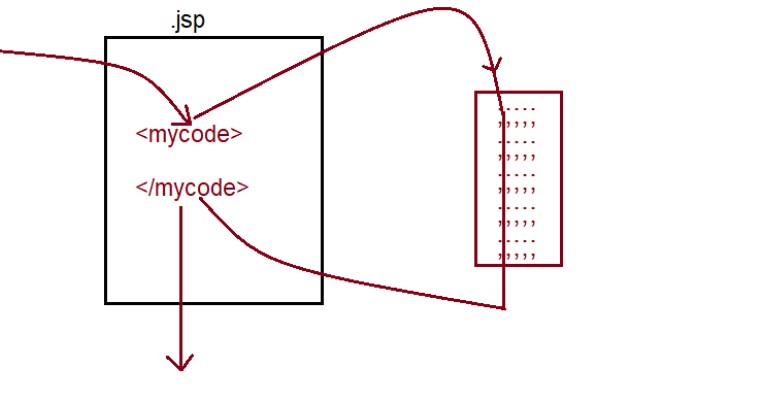
JSP Actions

In JSP technology, using scripting elements we are able to provide java code in JSP pages.

As per the theme of JSP writing java code is not allowed.

To eliminate java code from jsp page we need to use “JSP Actions”.

In JSP actions we provide scripting tag in JSP page and we provide code w.r.t Scripting Tag.



In JSP , it is recommended to not write a java code so what we do is we write the java code outside the JSP and put it inside the tag. And we need to keep the tag inside jsp in order to execute it.

**Note:-**

Whenever container encounter the Scripting tag, then container will execute respective code by this an action will be performed which is called JSP Actions.

In JSP we have two type of actions:-

🡺Standard Actions(Supplied by jsp technology only)

🡺Custom Actions(As per the user needs by taking the support of SRS we need to define our own)

Standard Actions

🡺<jsp:useBean>

🡺<jsp:setProperty>

🡺<jsp:getProperty>

🡺<jsp:include>

🡺<jsp:forward>

🡺<jsp:scriptlet>

🡺<jsp:expression>

🡺<jsp:declaration>

What is java bean?

It is normal java class with setter,getter defined for private variables of class.

To promote serialization for a java bean we use an interface called “Serializable”.

It is also called as “POJO”(Plan old Java object).

**<jsp:useBean>**

This JSP action is used to connect jsp with the java bean which contain setter and getter of JSP pages.

**<jsp:setProperties>**

This JSP Action is used to set properties in the JSP page

**<jsp:getProperties>**

This JSP Action is used to get the value of the element from the JSP page.

**<jsp:useBean id = "name of the reference " scope="[scopes of jsp]"**

**class="name of the class for which object should becreated:/>**

**X idvalue=(X)Class.forName([supplied value in class]).newInstance();**

Let’s understand by using Code:-

**<jsp:setPropertied property=” “name=” “value=” “/>**

**name.setPropertiesValue(Value supplied);//dynamic setting of a value**

**<jsp:getProperties property=”” name=””/>**

**name.getPropertyValue()**

JSP Page:-

<jsp:useBean id="student" class="in.ineuron.bean.Student" scope="page">

<jsp:setProperty property="id" name="student" value="10" />

<jsp:setProperty property="name" name="student" value="sachin" />

<jsp:setProperty property="address" name="student" value="MI" />

<jsp:setProperty property="age" name="student" value="49" />

</jsp:useBean>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">

<title>output</title>

</head>

<body>

<center>

<h1>STUDENT DETAILS</h1>

<table>

<tr>

<th>ID</th>

<td>

<jsp:getProperty property="id" name="student" />//property is instance variable and name is reference variable.

</td>

</tr>

<tr>

<th>Name</th>

<td>

<jsp:getProperty property="name" name="student" />

</td>

</tr>

<tr>

<th>Address</th>

<td>

<jsp:getProperty property="address" name="student" />

</td>

</tr>

<tr>

<th>Age</th>

<td>

<jsp:getProperty property="age" name="student" />

</td>

</tr>

</table>

</center>

</body>

</html>

Bean page

package in.ineuron.bean;

public class Student {

private Integer id;

private String name;

private String address;

private Integer age;

static {

System.out.println(".class file is loading");

}

public Student() {

System.out.println("Student object is created");

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

System.out.println("Student.getId()");

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

System.out.println("Student.getName()");

this.name = name;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

System.out.println("Student.getAddress()");

this.address = address;

}

public Integer getAge() {

return age;

}

public void setAge(Integer age) {

System.out.println("Student.getAge()");

this.age = age;

}

@Override

public String toString() {

return "Student [id=" + id + ", name=" + name + ", address=" + address + ", age=" + age + "]";

}

}

Let’s make the page more dynamic here bean file will be same we will only change html page and jsp page

JSP Page

<%!

Integer id;

String name;

Integer age;

String address;

%>

<%

id=Integer.parseInt(request.getParameter("id"));

name=request.getParameter("name");

address=request.getParameter("address");

age=Integer.parseInt(request.getParameter("age"));

%>

<jsp:useBean id="student" class="in.ineuron.bean.Student" scope="page">

<jsp:setProperty property="\*" name="student"/> //binding will happen automatically if all the property name are same with the html

</jsp:useBean>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html;charset=ISO-8859-1">

<title>output</title>

</head>

<body>

<center>

<h1>STUDENT DETAILS</h1>

<table border=1>

<tr>

<th>ID</th>

<td>

<jsp:getProperty property="id" name="student" />

</td>

</tr>

<tr>

<th>Name</th>

<td>

<jsp:getProperty property="name" name="student" />

</td>

</tr>

<tr>

<th>Address</th>

<td>

<jsp:getProperty property="address" name="student" />

</td>

</tr>

<tr>

<th>Age</th>

<td>

<jsp:getProperty property="age" name="student" />

</td>

</tr>

</table>

</center>

</body>

</html>

Html Page will be like this:-

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Registration Page</title>

</head>

<body>

<center>

<h1>Registration page</h1>

<form method ="get" action="./index.jsp">

<table>

<tr>

<th>ID</th>

<td><input type='text' name='id' placeholder='id' /></td>

</tr>

<th>Name</th>

<td><input type='text' name='name' placeholder='Name' /></td>

</tr>

<tr>

<th>Address</th>

<td><input type='text' name='address' placeholder='Address' /></td>

</tr>

<tr>

<th>age</th>

<td><input type='text' name='age' placeholder='age' /></td>

</tr>

</table>

<input type="submit" value="Reg"/>

</form>

</center>

</body>

</html>

What is the process here let’s see

🡺 First we will shoot the request from the html where the id ,name,age and address are the property name which is same to the jsp file

🡺The control will go to the jsp page where we have written declarative and scriptlet tag to parse the value as the value which is coming from the html are string we need to do parse the value in respective wrapper class.

🡺 We don’t need to set the value in the jsp:setProperty as the name of the property is same in html to name in the JSP.

As notice above inside html page name attribute values and in the bean instance variable(fieldname) are same

<input type ='text' name = 'id'/>

<input type ='text' name = 'name'/>

<input type ='text' name = 'age'/>

<input type ='text' name = 'address'/>

**class Student**

**{**

**Integer id;**

**String name;**

**Integer age;**

**String address;**

**}**

If both variable names are same, then instead of binding the value to each variable explicitly as shown below

<jsp:setProperty property="id" name="student" value='<%=id%>' />

<jsp:setProperty property="name" name="student" value='<%=name%>' />

<jsp:setProperty property="address" name="student" value='<%=address%>' />

<jsp:setProperty property="age" name="student" value='<%=age%>' />

we can bind it automatically using "\*"

**<jsp:setProperty property="\*" name="student" />**

**2.<jsp:include>=> include request Dispatching mechanism.**

**<jsp:forward>=> forward request Dispatching mechanism.**

**<jsp:param name = '' value=''"> => To add new values to request object and**

**send it to the respective page we use**

**<jsp:param name = ''value=''/>.**

**Let’s see how to connect database using jsp :-**

JSp page for UI

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<center>

<h1>Registration Form</h1>

<form method="get" action='./process.jsp'>

<table>

<tr>

<th>Name</th>

<td><input type='text' name='name' /></td>

</tr>

<tr>

<th>Address</th>

<td><input type='text' name='address' /></td>

</tr>

<tr>

<th>Age</th>

<td><input type='text' name='age' /></td>

</tr>

<tr>

<td><input type='submit' name='Reg' /></td>

</tr>

</table>

</form>

</center>

</body>

</html>

JSP page for processing the request

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8" import="java.sql.\*"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<%!String name;

String address;

Integer age;

//Resource for JDbc

static Connection connection = null;

static Statement stmt = null;

ResultSet resultset = null;

static {

try {

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

String url = "jdbc:mysql:///ineuron";

String username = "root";

String password = "Gaurav45";

connection = DriverManager.getConnection(url, username, password);

stmt = connection.createStatement();

} catch (SQLException ex) {

} catch (ClassNotFoundException e) {

}

}%>

<%

name = request.getParameter("name");

age = Integer.parseInt(request.getParameter("age"));

address = request.getParameter("address");

resultset=stmt.executeQuery("select \* from user where name='"+name+"'");

boolean available =resultset.next();

if(available){

%>

<jsp:forward page="existed.jsp" />

<%

}

else{

String sqlInsertQuery = String.format("insert into user(name,age,address)values('%s','%d','%s')",name,age,address);

int rowAffected=stmt.executeUpdate(sqlInsertQuery);

if(rowAffected==1){

%>

<jsp:forward page="./success.jsp" />

<%}

else{

%>

<jsp:forward page="./failure.jsp" />

<% }

}

%>

</body>

</html>

Customactions

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

These are the actions which are developed by developers as per their

application requirements.

In jsp two types of tags are available

a. standardactions -> predefined tags known to container

b. customactions -> inform explicitly to the container.

To prepare custom tags in Jsp pages we use the following syntax.

<prefix\_name:tag\_name>

;;;;;;

;;;;;;

;;;;;;

;;;;;;

</prefix\_name:tag\_name>

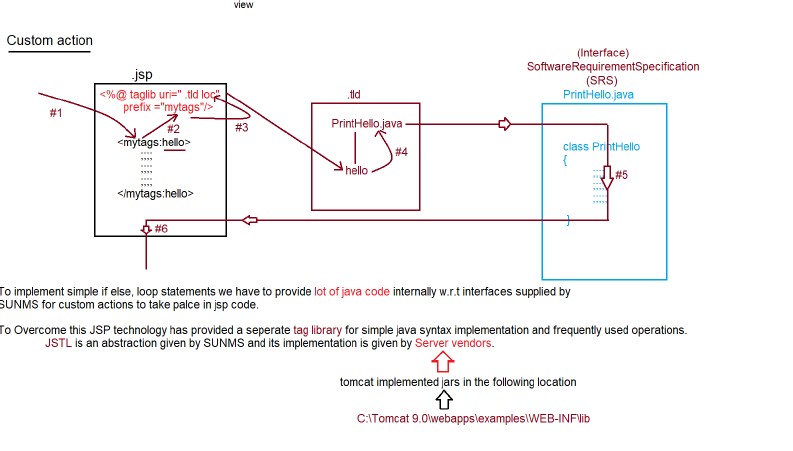
If we want to design custom tags in our jsp application, then we use the following

3 elements

a. jsp page with taglib directive

b. TLD file(Tag library descriptor)

c. Tag Handler class.



JSP🡺 To Write view (presentation layer) and to make page dynamic. Avoid writing java cide as much as possible.

EL(Expression Language) and JSP Actions🡺we can avoid java code, nut not able to replace all the functionality of java.

JSP Actions:-

a.Standard actions 🡺Tags are limited in number.

b.Custom action🡺user should write the tag and its working which is difficult and lengthy for programmer

JSTL(JSP standard Library)🡺 JSTL is comprised of five library

1.Core library(Commonly Used)

2.SQL library

3. Functional Library

4.Formating Library

5.XMl Library

To Use JSTL jar is supplied by tomcat vendor we refer to the following location:-

**C:\Tomacat9.0\webApps\examples\web-inf\lib**

**1.Core Library:-**

🡪**General purpose Tag** 🡪**Conditional Tag 🡪Iteration Tag 🡪URl Related tags**

<c:out> <c:if> <c:forEach> <c:import>

<c:set> <c:Choose> <c:forTokens> <c:url>

<c:remove> <c:When> <c:redirect>

<c:catch> <c:otherwise> <c:param>

**We need to add this tag lib before using JSTL:-**

**<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="Need to write the prefix of tag here"%>**

**Core Library:-**

**🡪**<c:out>

It is used for writing template text data and expression to the JSP. Just like the Sysout Statement.

Like

<c:out value=”Welcome to the JSTL Coding”/>

c=> prefixName

out=>tagName

tag lib tag for <c:out> will be

**<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>**

**<c:out value="welcome to JSTL coding.."/><br>**

**The user name is ::<c:out value="${param.user}"/><br>**

**The password is ::<c:out value="${param.password }"/><br>**

**To use expression expression tag inside JSTL we use “${ any expression}”**

**2.<c:set>**

We can use to set attribute in any scope and to set map and bean properties also.

<h1 style ='color:red;text-align : center'>

<c:set var="x" value="10" scope ="request"/>

<c:set var="y" value="20" scope ="request"/>

<c:set var="sum" value="${x+y}" scope ="session"/>

The result is::<c:out value="${sum}"/>

**3.<c:remove>**

To remove attribute in the specified scope we can use this tag.

If the scope is not specified for removing, by default it will search in

a.Page Scope

b.Request Scope

c.session Scope

d.Application Scope

**<c:remove var ="x"/>**

To remove attributes in the specified scope we can use this tag.

If the scope is not specified for removing, by default it will search in

a. Page Scope

b.request Scope

c.session Scope

d.application Scope

**eg::**

**<c:set var="x" value="10" scope ="page"/>**

**<c:set var="y" value="20" scope ="page"/>**

**<c:set var="z" value="${x+y}" scope ="session"/>**

**<h1>**

**The result is ::<c:out value="${z}"/>**

**</h1>**

**<c:remove var="x"/>**

**<c:remove var="y"/>**

**<c:remove var="z"/>**

**<h1>**

**The result is :: <c:out value="${z}"/>**

**</h1>**

**<c:catch>**

To catch the exception from the code we use <c:catch>

**Hibernate and JDBC**

🡺In a stand alone application there is three layers namely

a.Controller Layer

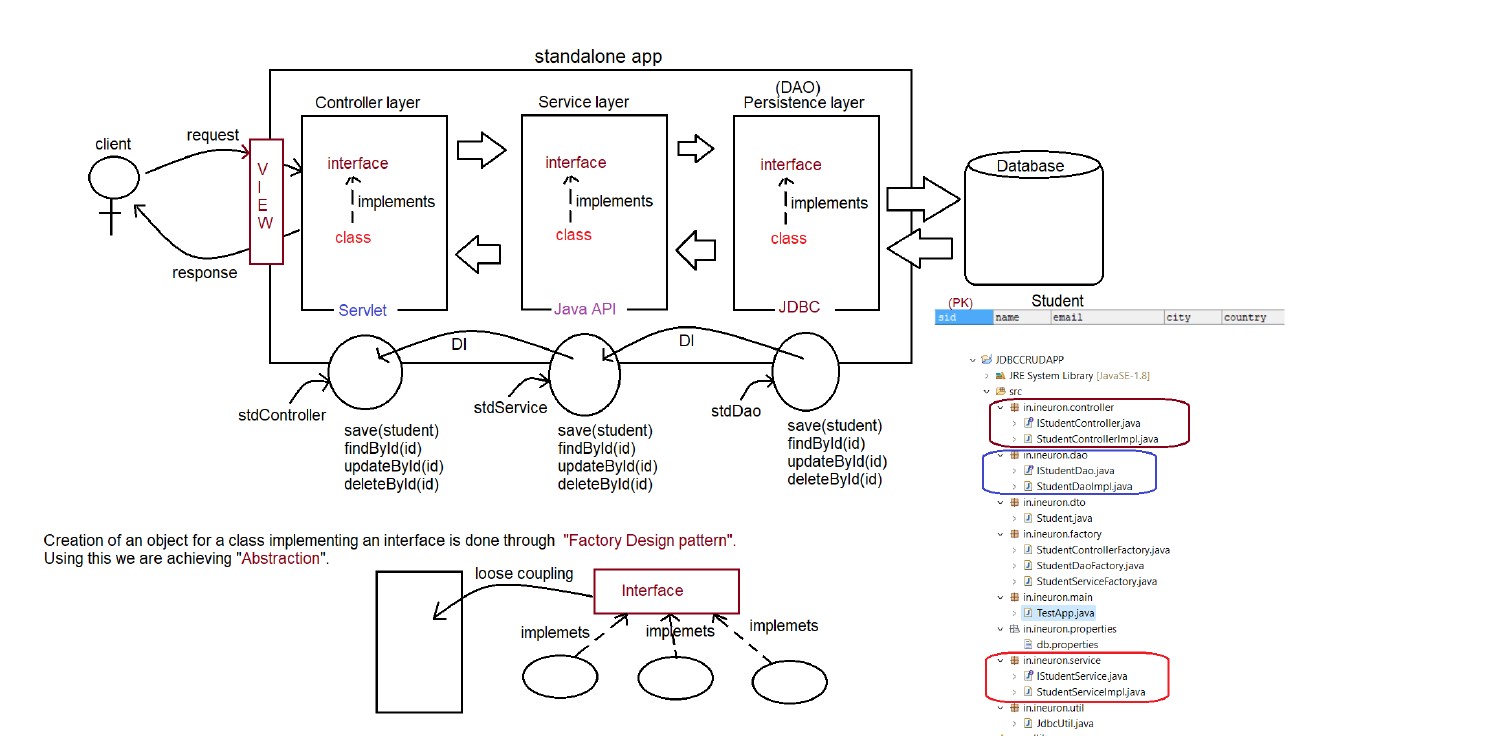
b.Service Layer

c.Persisistence layer

**Controller Layer** consist of servlet which do all the processing purpose for client request.

**Service Layer** consist of third party api like jstl and all

**Persistence layer** is layer which is responsible for connecting the application with the database.



**Logical database connection:-**

**JDBCUTIL file:-**

**package in.ineuron.util;**

**import java.io.FileInputStream;**

**import java.io.IOException;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.SQLException;**

**import java.util.Properties;**

**import com.zaxxer.hikari.HikariConfig;**

**import com.zaxxer.hikari.HikariDataSource;**

**//Using Hikaricp configuration for connection pooling**

**public class JdbcUtil {**

**private JdbcUtil() {**

**}**

**static {**

**try {**

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

**} catch (ClassNotFoundException e) {**

**e.printStackTrace();**

**}**

**}**

**public static Connection getJdbcConnection() throws SQLException, IOException {**

**HikariConfig config=new HikariConfig("E:\\workspaces\\myWorkspace\\JDBCCRUD1\\src\\main\\java\\in\\ineuron\\properties\\db.properties");**

**HikariDataSource dataSource=new HikariDataSource(config);**

**Connection connection =dataSource.getConnection();**

**return connection;**

**}**

**public static Connection physicalConnection() throws SQLException, IOException {**

**FileInputStream fis=new FileInputStream("E:\\workspaces\\myWorkspace\\JDBCCRUD1\\src\\main\\java\\in\\ineuron\\properties\\db.properties");**

**Properties properties=new Properties();**

**properties.load(fis);**

**String url=properties.getProperty("jdbcUrl");**

**String username=properties.getProperty("dataSource.user");**

**String password=properties.getProperty("dataSource.password");**

**Connection connection=DriverManager.getConnection(url,username,password);**

**return connection;**

**}**

**}**

**From xml file:-**

**jdbcUrl=jdbc:mysql://localhost:3306/ineuron**

**dataSource.user=root**

**dataSource.password=Gaurav45**

We need to add these dependencies in our project:-

<dependency>

<groupId>com.zaxxer</groupId>

<artifactId>HikariCP</artifactId>

<version>5.1.0</version>

</dependency>

<dependency>

<groupId>

org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.17</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.17</version>

</dependency>

**Persistence Logic**

The logic which is written to perform persistence operation is called persistence logic.

Operation are CRUD/SCUD/CURD

To Write persistence logic in java we have technology and framework

a.Technology:-JDBC

b.Framework:- ORM tool like hibernate,JPA, Spring ORM, Spring data JPA.

**Technology:-**

It is build on top of language

**Framework:-**

It is built on top of technology

When we already have jdbc as persistence logic, what is the need of ORM?

Limitations of JDBC

a. if we use jdbc to develop persisitence logic, we need to write sql queries by following the syntax of “Database” DB Queries are specific to Database,this makes JDBC not portable across multiple databases.

JAVA🡺WORA

JAVA+JDBC(Not Supported)

b. Jdbc technology if we use and write a code, there would be a boiler plate code in our application.

c. JDBC technology throws only one Exception called”SqlException”, but it is CheckedException which mean us should have handling the logic otherwise code would not compile.

d. JDBC tecnlogy has only Exception called SQLException , so we don’t have detailed hierarchy of Exceptions related to different Problems.

e.JDBC ResultSet object is not serializable , so we can’t send it over the network, we need to use bean/pojo to send the data over the network by writing our own logic.

f. While closing the jdbc connection object, we need to analyse the code allot otherwise it would result in “NullPointerException.

g. JDBC doesn’t have good support of transaction management

a.local

b.global(NO support of JDBC)

h. JDBC support only positional Parameters. It is difficult for the user to inject the values

it doesn’t support named parameters.

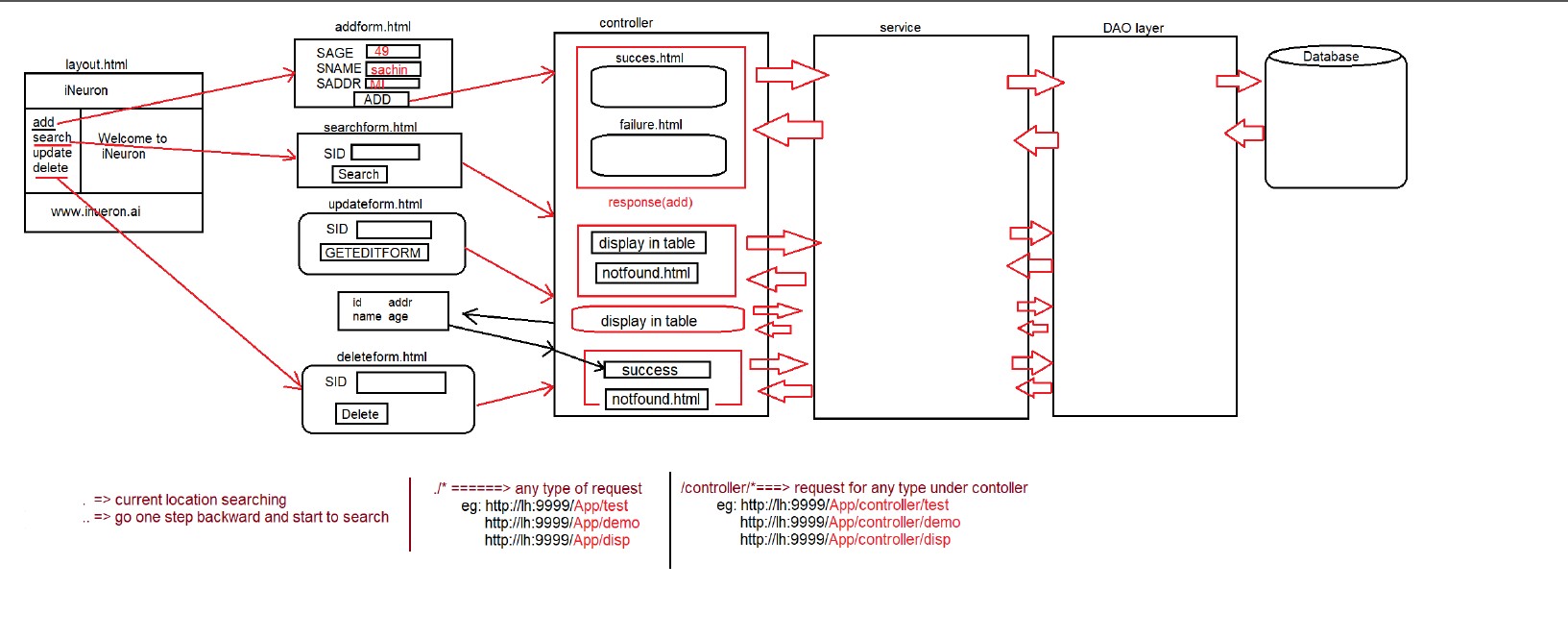
String SQLINsertQuery=”insert int o student(“name’,’email’,’city’,’country’) values(?,?,?,?)”;

i.To use JDBC, Strong Knowledge of SQL is required.

j. JDBC doen’t support versioning,timestamp as inbuilt feature.

Versioning-keep track of how many times record get modified

timestamp:-keep track of when record was inserted and when lastly it was modified.



The annotation @WebServlet("/controller/\*") is used in Java Servlet-based web applications to declare a servlet and map it to a specific URL pattern.

Breakdown:

@WebServlet: This annotation is used to declare a servlet without needing to define it in the web.xml file.

"/controller/\*": This is the URL pattern the servlet will respond to. The \* means that any URL that starts with /controller/ will be handled by this servlet.

In a Dynamic Web Project (like one created in Eclipse with a servlet container like Apache Tomcat), the default HTML (or JSP) page that gets executed when the app is run depends on how the project is configured.

By default, the welcome page is:

index.html or index.jsp

These are usually placed inside the WebContent or webapp directory (depending on your IDE or Maven setup).

.=> current location searching ./\*🡺Any type of Request /controller/\*🡺request for any type under

..=>go one step backward and start to search eg:http:lh:9999/app/test controller

<http://lh:9999/app/disp> eg:http//lh:9999/app/controller/test

eg:http//lh:9999/app/controller/demo

eg:http//lh:9999/app/controller/disp

**Hibernate**

To overcome the limitation of JDBC, we need to opt for ORM.

1.JDBC code is not portable.

2.JDBC we can’t deal with Object injection to the database as the query expects values.

ORM=> It stands for object Relation Mapping.

Entity🡪 Entity is table information present in the form of java object like

@Entity (SUNMS)

public Class Student{

JPA Specification

Integer sid;

String Sname; Implements Implements

String Saddresss;

}

HIbernate

Ibatis

ORM(ORM)🡪 One object of the Hibernate is map with one row of the table, it is map using XML/ Annotation

XML/Annotation Connection using XML/Annot

Configuration details like

a.URL

b.Username

c.Password.

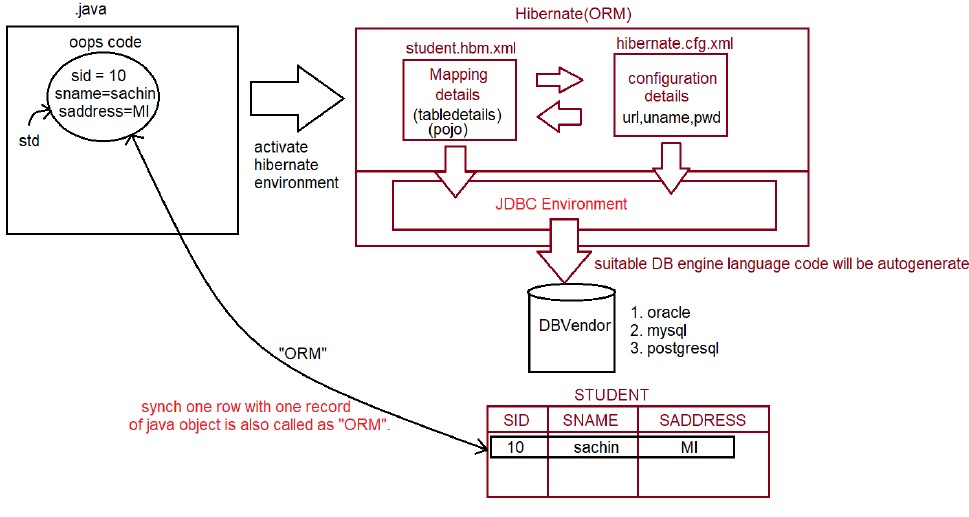
Student STUDENT

sid SID

sname SNAME

saddress SADDRESS

After connecting the Hibernate using XML or annotation and goes to the jDBC to execute the quesry because behind the scene Hibernate use JDBC to excute the query.



what is ORM?

It stands for Object relational mapping, where the programmer would map the table details with java object details

through xml/annotation.

JPA -> Java Persistence API (set of rules and guidelines to implement ORM)

Hibernate is a tool/framework implemented for JPA specification given by SUNMS.



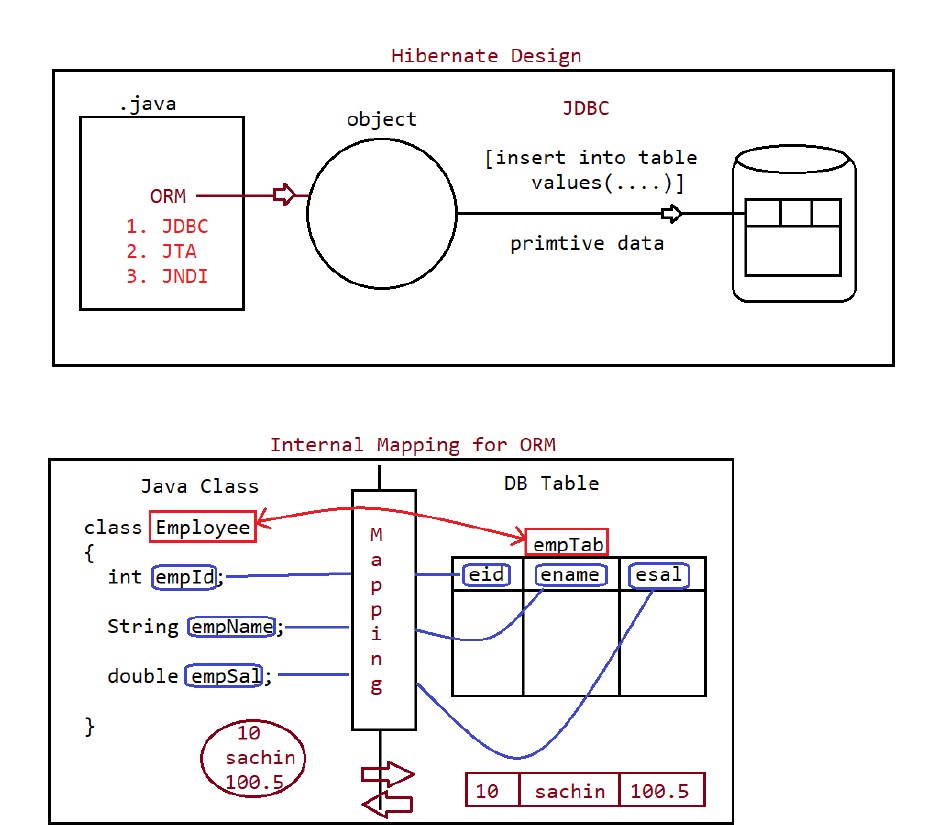
To Write one application we need to use 4 files as shown below:-

a. Model class

b.Mapping Code(XML/Annotation).

c.Configuration files

d.Test class



**Modle class**

It represent Model data, it can be also called as entity/POJO

It is a class which follow rules given by hibernate Framework.

a. Class must be a package statement

b.Class must be a public type.

c. No.of tables =NO. of Classes

d. Class can have variables, must be private

{No. of Column=No. of Variables}

e.

Class should have zero argument constructor and setter-getter methods.

f. Class can override toString(), hashCode(),equlas() from Object class.

g. Class can have annotations given by JPA and also core library annotations.

h. Class can inherit(IS-A)[extends/implements] only hibernate api.

**Mapping Code**

**1.** @Entity🡪 It maps model class with DB Table and variables with Column Names

**2.** @Id🡪 It indicates primary key, every table must contain primary key column.

**3.** @Table(Optional)🡪It indicates the table name which is been mapped with the Model class.

**4.** @Column(Optional)🡪 It indicates the clumn Name of table which is been mapped with Model class.

**Note:-**

If @Table ,@Colum are not provided then by default class Name is TableName, variable Name is Column Name(Taken by hibernate)

**Eg#1**

**@Entity**

**@Table(name=”empTab”)**

public class Employee{

**@Id**

**@Column(name=”eid”)**

private int empId;

**@Column(name=”ename”)**

private int empName;

**@Column(name=”esal”)**

private int empSal;

}

**Eg2**

**@Entity**

**@Table(name=”prodTab”)**

public class Product{

**@Id**

**@Column(name=”pid”)**

private int prodId;

**@Column(name=” pcode”)**

private int prodCode;

**@Column(name=”pcost”)**

private int prodCost;

}

**Mapping w.r.t XML**

**eg#1.**

<hibernate-mapping>

<class name="in.ineuron.model.Employee" table="empTab">

<id name="empId" column="eid"/>

<property name="empName" column="ename" />

<property name="empSal" column="esal" />

</class>

</hibernate-mapping>

**eg#2**

Product.hbm.xml

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

<hibernate-mapping>

<class name="in.ineuron.model.Product" table="prodTab">

<id name="prodId" column="pid"/>

<property name="prodCode" column="pcode" />

<property name="prodCost" column="pcost" />

</class>

</hibernate-mapping>

**Configuration File**

For one application, one configuration file should be given it is XML format.

Configuration=Properties+mapping class

Property=> it represents key-value pair.

**🡺**First we need to create the xml file with file name **hibernate.cfg.xml** and we need to add DTD(Document Type of Definition) of hibernate in the xml. We need to download the DTD for the particular version in order to tell the xml file for which purpose we are using the xml. Simply go the Google and find the DTD of particular version and add it into the xml file.

**<!DOCTYPE hibernate-configuration PUBLIC**

**"-//Hibernate/Hibernate Configuration DTD 3.0//EN"**

**"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">**

**🡺** Now we need to add **<hibernate-configuration> </hibernate-configuration>** inside this we need to write **<session-factory></session-factory>** tag where we need to add property tag. Inside property tag we need to define the driver name , url, username,password to connect the hibernate with java file.

**<property name="hdm2ddl.auto">update</property>**

This property is used to create table automatically if table is not available , if table is available then it only update the table.

As we know that hibernate will fire the query automatically and if want to know the query then we will use this property to know which query is fired from the hibernate.

**<property name="show\_sql">true</property>**

**XML file will be like this:-**

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE hibernate-configuration SYSTEM "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">**

**<hibernate-configuration>**

**<session-factory>**

**<!—Database connection Setting-->**

**<property name="connection.driver\_class">com.mysql.jdbc.Driver</property>**

**<property name="connection.url">jdbc:mysql://localhost:3306/myhiber</property>**

**<property name="connection.username">root</property>**

**<property name="connection.password">Gaurav45</property>**

**<!-- JDBC connection pool (use the built-in) -->**

**<property name="connection.pool\_size">1</property>**

**<!-- SQL dialect -->**

**<property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>**

**<property name="hbm2ddl.auto">update</property>**

**<!-- Echo all executed SQL to stdout -->**

**<property name="show\_sql">true</property>**

**<!-- Format SQLOuput to stdOut--->**

**<property name ="format\_sql">true</property>**

**</session-factory>**

**</hibernate-configuration>**

**<!-- SQL dialect -->**

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

dialect => It is a class available inside package called

org.hibernate.dialect,it will generate the SQLQuery when the programmer performs operation.

For every database dialect is different.

Oracle => nature of query

MySQL => nature of query

PostgreSQL => nature of query

**<!-- Echo all executed SQL to stdout -->**

<property name="show\_sql">true</property>

This property is used to see the Query generated by the dialect based on the datbase environment on the console.

**<!-- Echo all executed SQL to stdout -->**

<property name="format\_sql">true</property>

This property is used to fromat the Query generated by the dialect based on the datbase environment on the console.

**<property name ='hibernate.hbm2ddl.auto">[validate/create/update/create-drop]</property>**

**validate** =>hibernate creates no table, programmer should create or modify tables manually. this is considered as default value.

**create** = > hibernate always creates new table,if table exists it will drop the table.

**update** => hibernate creates new table,if table doesnot exists, otherwise it will reuse the same table.

**create-drop**=>This option is used for testing purpose not in development creates a new table and perform operation,at last it will drop the table.