

Author : Mr. SUMANTH

INDEX

Core Java Basics	2 - 27
JDBC	28 - 85
Servlets	86 - 129
JSP	130 - 145
Oracle	146 - 246
Hibernates	247 - 307
Springs	308 – 344
Struts	345 – 440

*****CORE JAVA BASICS*****

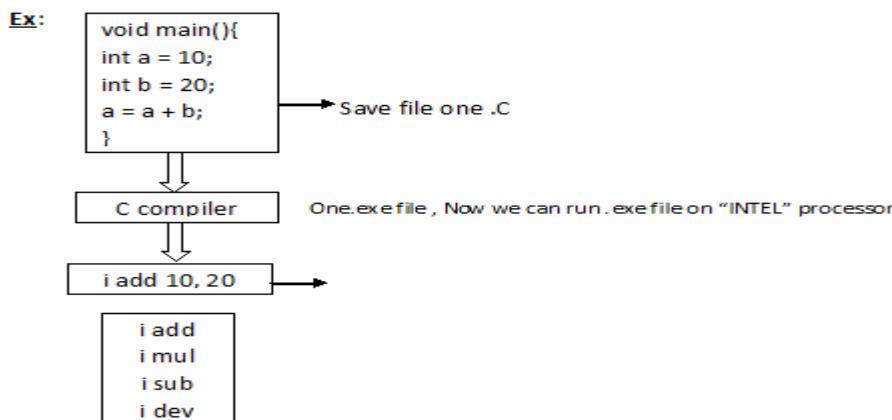
Basics:

There are so many companies who are manufacturing the processors, some of the company names are INTEL, SUN, AMD and etc. the **processors are responsible to execute the programs**. We have developed a ‘C’ program to add the numbers. The ‘C’ program developed high level language.

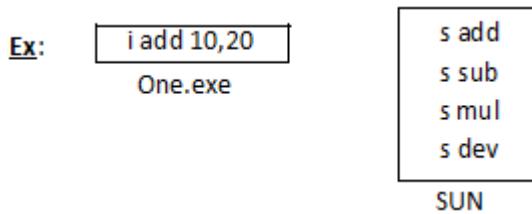
Ex:

```
void main(){  
    int a = 10;  
    int b = 20;  
    a + b = a;  
}
```

If we would like to run this program. We need to compile the program. **It is the responsibility of compiler to convert High level language into machine level language**. The ‘C’ compiler generates an .exe file. Inside the exe file the ‘C’ compiler has placed the processor dependent instructions. This ‘C’ compiler is installed on INTEL processor.



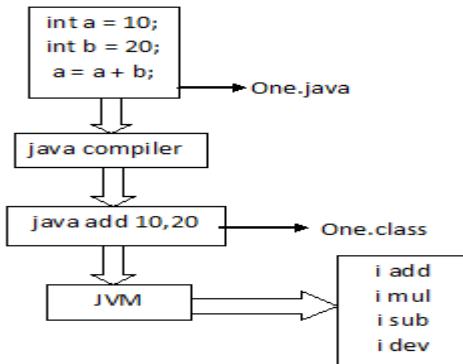
We have taken the same .exe file and try to run it on sun processor, but we are failed to execute the program. This is because sun processor cannot execute the “INTEL PROCESSOR” instructions.



If we got the requirement to run the ‘C’ program on all the processors we need to recompile the same program on different processor.

When develop a JAVA program to add the numbers we can not run the directly on any computer.

By taking the help of JAVA compiler we have converted .java programming to .class file. If we would like to run the java application we need to give .class file as Input to JVM. It is the responsibility of JVM to convert java instructions into processor dependent instructions.



To see what is available in a .class file, we can take the help of java profiler command. When we use the **javap** command we have observe the following code in Add.class.

Ex: Public class Add extends java.lang.object{
 public Add();
 public static void main(java.lang.String[]) //save Add.class

If we want to see all the instructions also we have to use an option “-c”

Ex: Javap -c Add ↪

- **We can invoke (call) the JVM by using the command java.** When we invoke the JVM it searches for appropriate .class file. If the class file is available **Class Loader** loads the class into JVM’s memory. If the class is not available JVM throws an Exception `java.lang.NoClassDefFoundError: sub`.

After the class is loaded JVM checks for main method. If it is available JVM starts the execution from main method. If the main method is not available JVM throws an Exception `NoSuchMethodFoundError: main`

```

Administrator: C:\Windows\system32\cmd.exe
E:\JavaProgs\Core>javac Cone.java
E:\JavaProgs\Core>java cone
Exception in thread "main" java.lang.NoClassDefFoundError: cone <wrong n.
at java.lang.ClassLoader.defineClass(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:791)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:449)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:449)
at java.net.URLClassLoader.access$100(URLClassLoader.java:71)
at java.net.URLClassLoader$1.run(URLClassLoader.java:361)
at java.net.URLClassLoader$1.run(URLClassLoader.java:355)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:354)
at java.lang.ClassLoader.loadClass(ClassLoader.java:423)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:308)
at java.lang.ClassLoader.loadClass(ClassLoader.java:356)
at sun.launcher.LauncherHelper.checkAndLoadMain(LauncherHelper.j
E:\JavaProgs\Core>java Cone
Hai
  
```

➤ When the complier will add the constructor?

When the programmer provides constructor in the java programmer the compiler will not add default constructor other wise compiler will Add default constructor.

```
Ex:    public class Add{  
        public Add(){  
        }  
    } // save Add.java
```

- When the compiler will add the code to inherit properties of java.lang.Object?

When the class is not Inheritance the properties of any other class then the compiler will try to inherit the properties of `java.lang.Object` class

In the following code the compiler try's to inherit the properties of java.lang.object

Ex: public class Cone { } // save Cone.java

In the following example java compiler will not inherit the properties of `java.lang.Object`

Ex: Public class Ctwo extends Cone { } //save Ctwo.java

We can compile and run the java programs using the following commands :

Compile : javac -g programname.java

Run : `java -Djava.compiler=NONE programname (or)`

java -Djava.compiler=none programname (or)

java –Djava.compiler programname

When ever we call the JVM , operating system loads the JVM program into RAM. Now the operating system allocates default memory (256mb) to JVM in a RAM.

EX:



After the program execute is finished, memory will be cleared from the RAM.

As start of the JVM we have a program class loader. **It is the responsibility of a class loader to load the class into JVM's memory.**

When ever JVM's starts the execution of a method it allocates two memory blocks. They are 1) Stack and 2) Method area

Stack is mainly used to perform any operations. Stack is a temporary memory location. If we want to store the data permanently, we store the data in method area.

JVM's removes stack, method area after finishing the execution of method.

This is the Java Instructions for “Add.class” . Command is **javap -c add**

```
public class Add extends java.lang.Object{  
    public Add();  
    code:
```

```

0: aload_0
1: invoke special #1;//Method java/lang/Object."<init>"()
4: return
public static void main(java.lang.String[]);
code:
0: bipush 10
2: istore-1
3: bipush 20
5: istore-2
6: iload-1
7: iload-2
8: iadd
9: istore-1
10: return
}

```

They are 3 types of variables in java.

- 1. Local variables**
- 2. Instance variables**
- 3. Static variables**

➤ The variables which are declared inside a method are called as local variables.
The method parameters are also called as local variables.

Ex:

```

public class One{
int a;
    } //Instance Variables
int b; } Static Variables
static int aa;

public void methodOne(int d, int e){
int x;
    } //Local Variables
int y;
}
}

```

The memory for the local variables will be allocated in stack.

The variables which are declared in a class are called as Instance variables. In the above examples the variables 'a' and 'b' are called as Instance variables , the memory will be allocated for the Instance variables in heap memory.



We have developed a class Cone with two Instance Variables as shown below.

Ex: public class Cone {
 int a;
 int b;
} //Cone.java

We have developed a reusable components Cone. Any body can use it by creating object to Cone class.

Ex: Public class MyApp {
 Public static void main(String[] args) {
 Cone c1 = new Cone();
 }
} // MyApp.java

When we execute the above java program it will divided creation of object line into two parts, they are

Cone c1;	part1 (c1 = Reference Variable)
c1 = new Cone();	part2 (c1 create object to Cone class)

When the part1 line is executed JVM allocates memory for c1 Reference Variable inside the Stack (c1 Reference (local variable)).

When part2 is executed JVM try to create object to Cone class.

While try to create Cone class performs the following steps.

1. **JVM checks for Cone.class** and try to load Cone.class files into JVM's memory.
2. Now JVM **opens Cone.class** and **check for no of Instance variables** in a Cone.class
3. Now the **JVM calculates the required memory for the Instance variables**.
4. Now the **JVM's checks where the suficient free space** is available to allocated the memory **for instance variables in JVM's Heap memory**.
5. After the memory is allocate , **JVM place the starting address of object into c1 Reference variable**.

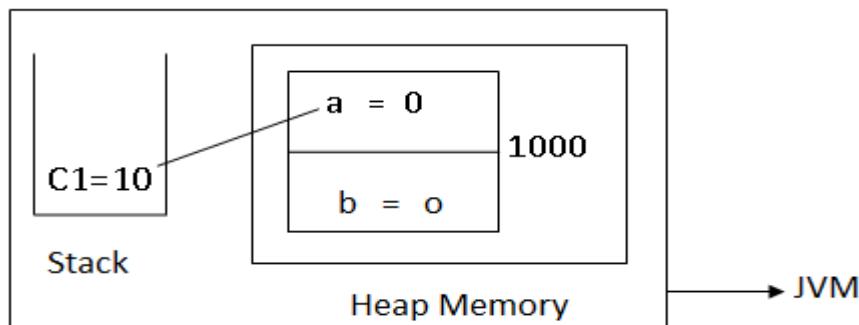
Note:

As the c1 variable is hold the address of object of Cone,then we call c1 as Reference variable.

When ever we create object or Instance for ever copy it contains one copy of variables. In java all the local variable must be initializing with some values, without initialize the local variables when we try to compilation a program we get a compilation error.

```
Ex:    Public class MyApp{  
        Public static void main(String[] args){  
            int a;  
            System.out.println(a);  
        }  
    } // MyApp.java
```

All the instance variables are initializing with default values when the object is created.



We can access methods of a class based on reference variables or objects.

```
Ex: public class Cone {  
    int x;  
    int y;  
    public void methodOne() {  
        System.out.println("we are Cone methodOne");  
        System.out.println("x value is: " +x);  
        System.out.println("y value is:" +y);  
    }  
} // Cone.java
```

```
public class MyApp {  
    public static void main(String[] args) {  
        Cone c1 = new Cone();  
        Cone c2 = new Cone();  
        c1.X = 1;  
        c1.y = 2;  
        c2.x = 11;  
        c2.y = 22;  
        c1.methodOne();  
        c2.methodOne();  
    }  
}
```

```

    }
}           // MyApp.java

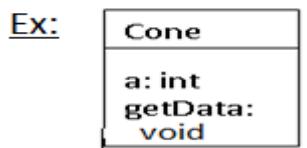
```

Generally the team leaders of responsible to develop the UML diagrams.

Once the UML diagrams release we are responsible to convert UML diagrams to corresponding java program.

Note: UML diagrams are used for any programming language.

The following is an example class diagram.



The compiler compiles all the dependent class of a java program.

Ex: Public class MyApp{

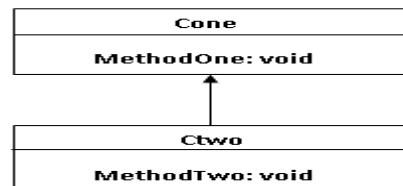
```

    Public static void main(String[] args){
        Cone c1 = new Cone();
        Ctwo c2 = new Ctwo();
        C.methodOne();
    }
}

```

When we compile the java program the java compiler compile three different programs they are Cone.java, Ctwo.java and MyApp.java

The team leader has given the UML diagrams.



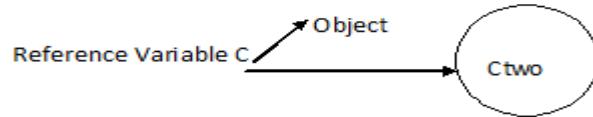
```

public class Cone {
    void methodOne() {
        System.out.println("we are in Cone method()");
    }
}           //Cone.java

public class Ctwo{
    void methodTwo(){
        System.out.println("we are in Ctwo methodTwo()");
    }
}           //Ctwo.java

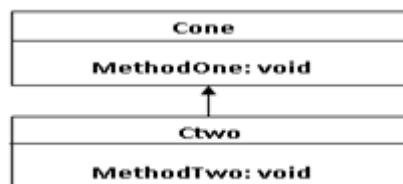
```

It is always recommended to create object to sub class. By using sub class object we can access the methods of that class as well as super class of that class.



From the above diagram we have understand that 'C' is reference variable is hold in Ctwo class.

*The following is a UML diagram given by the leader.



Now we have developed two classes Cone and Ctwo.

```

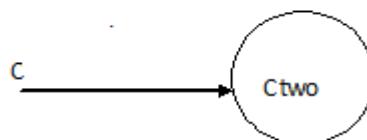
public class Cone{
void methodOne(){
System.out.println("we are in Cone methodOne()");
}
}

public class Ctwo extends Cone{
void methodTwo(){
System.out.println("we are in Ctwo methodTwo()");
}
}

public class MyApp{
public static void main(String[] args){
Ctwo c = new Ctwo();           Step - 1
c.methodOne();                Step - 2
}
}                                // MyApp.java

```

When JVM execute it creates reference variable C and Ctwo class object.



When JVM executes Step-2 JVM executes C reference variable is holding which class object JVM finds that C reference variable is holding Ctwo class object.

Now the JVM open Ctwo class object and check method one() is available as part of Ctwo class or not. If available in Ctwo class JVM executes it. If it is not available JVM checks whether Ctwo class is inheriting the properties of any other class.

In our example Ctwo class is inheriting the properties of Cone class. Now the JVM open's Cone class and check whether methodOne() is available in Cone class. JVM executes methodOne() Cone class.

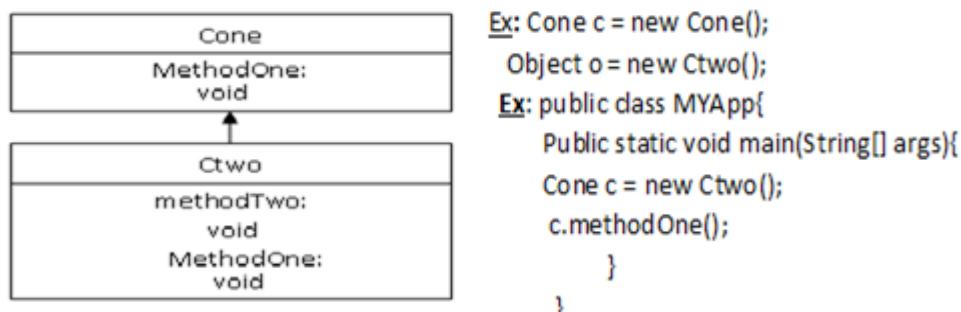
In **java** we can achieve polymorphism by using inheritance.

What is polymorphism?

An object which exhibits multiple behaviors is called as polymorphism.

In java a super class reference variable can hold sub class object.

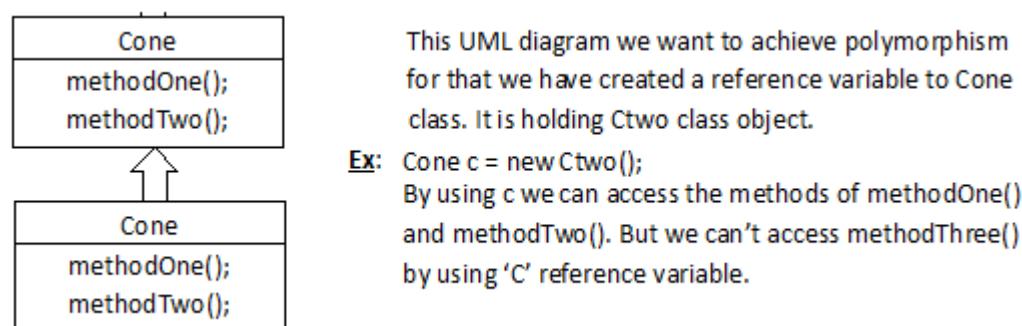
According to the above UML diagram the following code is valid.



When we got an error message like javaC is not recognized as internal or external commands we can resolve the problem by setting path and environment variable in the command prompt.

Ex: Set PATH=c:\program files\java\jdk1.6_29\bin

If a super class reference variable can holds sub class object, by reference variable we can access the methods of that class or super class.



A reference variable can hold an object. By using a method getClass() we can find which class object is holed by the reference variable.

Ex: Cone c1 = new Cone();

```
System.out.println(c1.getClass());
```

A reference variable can hold the address of another object.

Ex:

```
public class MyApp{
    public static void main(String[] args){
        Cone c1 = new Cone();
        Cone c2 = new Cone();
        c1.x = 10;
        c1.y = 20;
        c2.x = 60;
        c2.y = 70;
        System.out.println(c1.x);
        System.out.println(c1.y);
        System.out.println(c2.x);
        System.out.println(c2.y);
        c1 = c2;
        System.out.println(c1.x);
        System.out.println(c1.y);
        System.out.println(c2.x);
        System.out.println(c2.y);
    }
}
```

A super class reference variable can hold sub class object. By using the super class reference variable we can access the methods of that class only.

But we can't access the specific methods of sub class , to access the specific methods of sub class **we must type cast super class reference variables into sub class.**

Ex:

```
public class MYApp {
    public static void main(String args[]){
        Object o = new Ctwo();
        Cone c1 = (cone)o;
        Ctwo c2 = (ctwo)o;
        c2.methodTwo();
    }
}
```

We have developed Cone.java with two methods?

```
public class Cone{
    public void methodOne(){
        System.out.println("we are in Cone methodOne()");
        this.methodTwo();
    }
}
```

```
public void methodTwo{
System.out.println("we are in Cone methodTwo()");
}
}
```

We have developed Ctwo class by overriding methodTwo() in it?

```
public class Ctwo extends Cone{
public void methodTwo(){
System.out.println("we are in methodTwo()");
}
}
```

In the main() we have created object to Ctwo class and call methodOne()

```
public class MyApp{
public static void main(String[] args){
Ctwo c = new Ctwo();
c.methodOne();
}
// Step 1
}
```

When we execute the above program JVM checks which class object is held by the reference variable 'C'.

Now the JVM opens that reference variable and check for methodOne(). In our example 'C' reference variable is holding Ctwo class object.

Now the JVM checks methodOne() is available in Ctwo class or not. If it is not available JVM has checks the super class Cone. Now the JVM executes methodOne() of Cone class.

The internal code of methodOne() calling methodTwo(). Now the JVM checks whether methodTwo() is available in the current object or not , in our example in current object is Ctwo so the JVM open Ctwo class object and check for methodTwo(). If it is available it will execute or it goes to super class of Cone and check for methodTwo().

Interfaces: An interface can contain set of abstract methods. In an interface we cannot provide method with body. By default all the methods in an interface are "Abstract".

As part of an interface we can declare the variable, **when we declare variables in the interface, by default they are "static and final" variables.**

Ex: public interface MyInterface{
int a = 10;
public void methodOne();
}

We cannot create an object to an interface?

```
MyInterface m = new MyInterface(); // Invalid
```

We can create reference variable to an interface?

```
MyInterface m;
```

Once the interface is released, any body can provide the implementation to an interface. Providing implementation means writing the method body.

The final variable values will not able to modify. We can assign the value to final variable only once. (at a time)

Once if we declared final, no body will be able to override in the sub class.

If we declared final classes no body will be able to inherit properties of final classes.

String class is the final class this is because no body could like to inherit properties of string class.

for example some body as created the following MyInterface with two abstract methods.

```
public interface MyInterface{  
    public void methodOne();  
    public void methodTwo();  
}
```

// MyInterface.java

If we would like to provide the implementation to the above interface we must provide the implementation to all the abstract methods, other wise java complier will not allow us to compile the program.

Ex:

```
public class MyImp implements MyInterface{  
    public void methodTwo(){  
    }  
}
```

//MyImp.java

When we compile the above java program we get a compilation error MYImp is not abstract and does not override all the abstract methods().

To an interface we can create the reference variable the reference variable holds an object of a class which provides the implementation of a interface.

```
public class MyApp{  
    public static void main(String[] args){  
        MyInterface m;  
        m = new MyImp();  
        m = methodOne();  
        m = methodTwo();  
    }  
}
```

```
    }
} // MyApp.java
```

In a class we can provide abstract methods. If we provide abstract methods we must be declared that class as an abstract.

Ex: abstract class Cone{
 public void methodOne(){
 }
 public void methodTwo(){
 }
 // Cone.java

We cannot create object to any abstract class. We can create only reference variable.

We can declare of abstract if we don't allow any body to create object to a class.

Ex: public abstract class Cone{
 public void methodOne(){
 }
 Public abstract void methodTwo();
}

To the above Cone class we cannot create the object.

```
Cone c = new Cone(); // invalid
```

For an abstract class we can create only reference variable. i.e.

```
Cone c;
```

Once if we have an abstract class any body can inherit the properties of abstract class.

When they inherit properties of abstract class, they must provide the implementation to all the abstract methods.

Ex:

```
public class Ctwo extends Cone{  
    public void methodTwo(){  
        }  
}
```

We are training to a project to calculate electricity bill as well as water bill. The following programs calculate electricity bill as well as water bill **with out polymorphism**.

```
Public class ElecBill {  
    public void CalcElecBill(int units){  
        double bamount = units*2;  
        System.out.println("ElecBill amount is:" + bamount);  
    }  
} // ElecBill.java
```

```
public class WaterBill{
```

```

public void CalcBill(int units){
    double bamount = units*0.50;
    System.out.println("water bill amount is:" +bamount);
}
} // WaterBill.java

public class APBill{
    public static void main(string[] args){
        ElecBill e = new ElecBill();
        e.calElecBill(100);
        WateBill w = new WaterBill();
        w.calWaterBill(200);
    }
} // APBill.java

```

In the above ApBill.java, we are calculating ElecBill by using an object 'e' we are using 'w' to calculate WaterBill.

As we are using two different objects to calculate ElecBill as well as WaterBill, we cannot call this program as polymorphism.

The following example demonstrates how you achieve polymorphism for the same above project. **To achieve polymorphism we take the help of interface** will force every developer to provide the implementation to that interface.

```

public Interface Bill{
    public void calBill(){
    }
}

public class ElecBill implements Bill {
    public void calBill(int units) {
        double bamount = units * 2;
        System.out.println("ElecBill amount is:" + bamount);
    }
} //ElecBill.java

public class WaterBill implements Bill{
    public void calBill(int units){
        double bamount = units *0.50;
        System.out.println("WaterBill amount is:" + bamount);
    }
} //WaterBill.java

public class APBill{

```

```

public static void main(String[] args){
    Bill b;
    b = new ElecBill();
    b.calBill(100);
    b = new WaterBill();
    b.calBill(100);
}
}                                //APBill.java

```

In the above APBill.java the object ‘b’ is used to calculate both ElecBill as well as WaterBill. Because of this reasons we calAPBill.java as polymorphism code.

The following example demonstrates use of abstract class.

```

public interface Car{
    public void ty();
    public void eng();
    public void st();
    public void br();
}
}                                //Car.java

```

MRF guis are interest to provide implementation only some methods of the car interface.

```

public abstract class MRF implements car{
    public void ty(){
        System.out.println("MRF Types");
    }
}
}                                //MRF.java

```

MRF guys are any other fellows how want to a manufacture a car they engine the properties of MRF as shown below.

```

public class Mar extends MRF{
    public void eng(){
        System.out.println("Mar using suzi");
    }
    public void st(){
        System.out.println("Mar using power Streeing");
    }
    public void br(){
        System.out.println("mar using drum braks");
    }
}

```

```
 }  
 } //Mar.java
```

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

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

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

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

Step2: Create the following folders in the above created project.

```
src = browser files  
bin\classes-class files  
lib-library files (jar files)  
doc-documentation  
tmp-temporary
```

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

Ex: Javac -d <destination-folders> sourcefileName

```
Javac -d d:\work\bms\bin welcome.java
```

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

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

Package domain.companyname.projectname.module (Or)

Package companyname.projectname.modulename

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

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

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

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

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

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

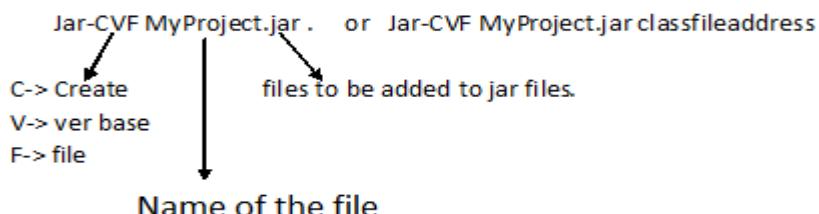
```
D:\> javac -d
```

*Procedure to create JAR file:

Step1: Goto a folder where all .class files are available use the following command to create JAR file.

```
Jar -cvf jarfilename files-to-be-added
```

Ex:



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

```
Jar -XVF MyProject.jar
```

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

Inside this folder we found an extra file MANIFEST.MF

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

The following are examples of creating a WAR file and EAR files.

```
Jar -CVF MyProject.war
```

```
Jar -CVF Myproject.ear
```

The main advantage of java is reusability. In java we can reuse the classes developed by somebody by creating the objects. If we would like to call the projects we must aware of all the methods of that class.

The **javadoc** command will help us increasing the doc as per the documentation we can find all the field summary, constructor summary & method summary.

The following is an example of using java doc command.

```
Javadoc Cone.java
```

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

```
Javac -d D:\work\lic\bin or D:\work\lic\src\*.java
```

```
bin> java info.inetsolv.jee.mb.welcome
```

```
delivering project.jar file
```

```
cd work\lic\bin>jar -cvf lib.jar
```

```
\bin>cd src
```

```
>src> java doc -d
```

```
D:\work\lic\doc welcome.java
```

*Environment Variables:

These are the variables which will be created for us to create Environment variable.

We use keyword set.

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

To see all the environment variables of our computer we use a keyword “set”.

To check the value of environment variable we use a command echo

Ex: echo% one %

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

Set one = 1;2;3 one is Variable name

Echo % one %

1;2;3

C:\> set one = 1

E:\> echo % one %

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

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

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

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

Set one = 0; % one %

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

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

Scope:

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

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

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

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

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

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

```
\bin>jar -CVF MyProject.jar  
C:> set CLASSPATH=C:\MyProject.jar;;  
C:> java info.inetsolv.jee.mb.welcome  
package info.inetsolv.jee.mb  
javac -d d:\work\bin  
D:\work\lic\src *.java  
java info.inetsolv.jee.mb.welcome  
set CLASSPATH=D:\work\lib\bin;;  
java info.inetsolv.jee.mb.welcome
```

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

Ex:

```
C:> set CLASSPATH=C:\MyProject.jar;; .cmd file  
C:> java info.inetsolv.jee.mb.welcome } b (or)  
C:>pause } sh for unix
```

To release project to customers we create jar file if we give only jar file customer is responsible to set the CLASSPATH and type the commands to run the project. To resolve this problem we develop a CMD file or data file. In this we supply all the DOS commands.

```
set CLASSPATH=c:\MyProject.jar;;  
java info.inetsolv.jee.mb.welcome  
pause //run.cmd  
creating an object's INSTANCE
```

***static variables:**

In a class we can declare static variables , **for the static variables memory will be allocated only once. This allocation will happen at the time of loading the class into JVM's memory.**

When ever we create object the class is loaded into JVM's memory.

```
Public class Cone{  
static int a;  
public void methodOne(){  
a = 10;  
}  
Static{  
System.out.println(a);
```

}

Memory Allocation

Local → Stack

Instance → Heap (one copy is created) variables

Static → Memory is not allocated to every object only one copy is allocated.

When class is loaded in JVM's memory, memory is allocated.

```
public class Cone{  
    static int a;  
    public static void methodOne(){  
        a = 10;  
    }  
    Static {  
        System.out.println(a);  
    }  
}
```

When we run above Cone.java by using java Cone, JVM loads Cone.class into JVM's memory.

When the class is loaded **JVM checks for static variables JVM allocates the memory for all the static variables and these variables are initialized with default values.**

Now the JVM checks are there any **static methods are available. If they are available it will allocate memory to all the static methods.** How the JVM checks are there any **static blocks are available. If they are available JVM allocates memory to static blocks.**

Order of Execution is Static Variable → Static Methods → Static Blocks

Now, the JVM checks are there any static blocks are available. If they are available JVM Executed static blocks. When ever we run a program JVM checks for public static void main() method. If it is available it starts the Execution from main()

```
public class Cone{  
    int a;  
    static int a;  
    static {  
        System.out.println("we are in Cone");  
    }  
} // save Cone.java
```

```
public class MyApp{  
    public static void main(String[] args){  
        Cone c1 = new Cone();
```

```
Cone c2 = new Cone();
}
} // save MyApp.java
```

Memory for static variables will be allocated only once when class is loaded into JVM's memory.

```
public class Cone{
int a;
static int b;
static{
System.out.println("we are in Cone");
}
} // save Cone.java
```

```
public class MyApp{
public static void main(String[] args){
Cone c1 = new Cone();
c1.a = 10;
Cone.b = 99;
System.out.println(c1.a);
System.out.println(Cone.b);
}
} // save MyApp.java
```

```
public class MyApp{
public static void main(String args[]){
Cone c1 = new Cone();
Cone c2 = new Cone();
c1.a = 10;
c1.b = 99;
c2.a = 20;
c2.b = 70;
System.out.println(c1.a);
System.out.println(c1.b);
System.out.println(c2.a);
}
} // save MyApp.java
```

```
public class MyApp{
```

```
public static void main(String[] args){  
    Cone c1 = new Cone();  
    Cone c2 = new Cone();  
    c1.a = 10;  
    c1.b = 99;  
    c2.a = 20;  
    System.out.println(c1.a);  
    System.out.println(c1.b);  
    System.out.println(c2.a);  
    System.out.println(c2.b);  
}  
} // save MyApp.java
```

*Hard coding:

Fixing the value to a variable is called as hard coding.

Ex: public class MyApp{
 public static void main(String[] args){
 int a = 10;
 System.out.println(a);
 }
 } // save MyApp.java

In the above program ‘a’ variable value is hard coded with a value ‘10’.

When we remove the hard coding project will be flexible. In a project it is not at all recommended to hard code the values we can use any of the following two techniques to remove hard coding. They are two types

1. Command line arguments
2. System properties

*Command line arguments:

```
public class MyApp{  
    public static void main(String args[]){  
        String color = ar[0];  
        int font = Integer.parseInt(ar[1]);  
        System.out.println(color);  
        System.out.println(font);  
    }  
} // save MyApp.java
```

To run the above java program we are using the following command.

```
>java MyApp blue 13
```

The disadvantage of above approach is if we inter change the values the project will be failed to resolve this problem. We use System properties.

*System properties:

```
public class MyApp{  
    public static void main(String[] args){  
        String color = System.getProperty("c1");  
        int font = Integer.parseInt(System.getProperty("f"));  
        System.out.println(color);  
        System.out.println(font);  
    }  
}
```

To run the above program we use the following command.

Ex: >java -D cl = green -DF =12 MyApp

Define Value
System variable name

A method which returns an object's factory method.

It's always recommended to remove hard coding.

If we remove hard coding project will be very flexible.

*class.forName():

Static method to class

It is used to load the class into memory without creating the object.

Ex:

```
public class MyApp{  
    public static void main(String[] args){  
        try{  
            Class.forName("Cone");  
        }  
        catch(ClassNotFoundException e){  
            System.out.println("class not available");  
        }  
    }  
}
```

All ways recommended handling the checked Exception by using any of the following two techniques.

1. try catch block
2. by using throws

When ever we use `Class.forName()` to load the class we must specify fully qualified name or absolute class name (class name with package).

Ex:

```
public class MyApp{
    public static void main(String[] args) throws ClassNotFoundException{
        Class.forName("java.lang.String");
    }
}
```

`Class.forName` return `Class` object this object contains two information they are

1. Name of the loaded class and its package.
2. We can get this information by using a methods like `getName()` `getPackage()`;

Ex:

```
class MyApp{
    public static void main(String[] args) throws ClassNotFoundException{
        Class c = Class.forName("java.lang.Object");
        System.out.println(c.getName());
        System.out.println(c.getPackage());
    }
}
```

`getPackage` method returns null value if the class does not contain package.

Develop a java program to create object to a class without using new operator.

***Using newInstance Operator:**

The following java program creates object to `Cone` class without using new operator and able to call methods.

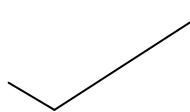
```
public class MyApp{
    public static void main(String[] args) throws Exception{
        Class c = Class.forName("Cone");
        Object o = c.newInstance();
        Cone c1 = (Cone)o;
        c1.methodOne();
    }
}
```

Ex:

```
public class Cone{ . . . }
```



```
public class Cone{
    public Cone(){
    }
}
```



```
public class Cone{  
    public Cone(int dummy){  
        }  
}
```



```
public class Cone{  
    public Cone{  
        }  
    public Cone(int dummy){  
        }
```



The following java program creates the object to any class dynamically.

```
public class MyApp{  
    public static void main(String[] args) throws Exception{  
        Class c = Class.forName(args[0]);  
        Object o = c.newInstance();  
        System.out.println(o.getClass());  
    }  
}
```

New instantiation method create object to a class if the class contains default constructor().

If class contains only parameter constructors() class.forName() fails in create a the object.

JDBC

*API (Application Programming Interface):

API is a document. (not a software)

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

1. Public API
2. Proprietary API

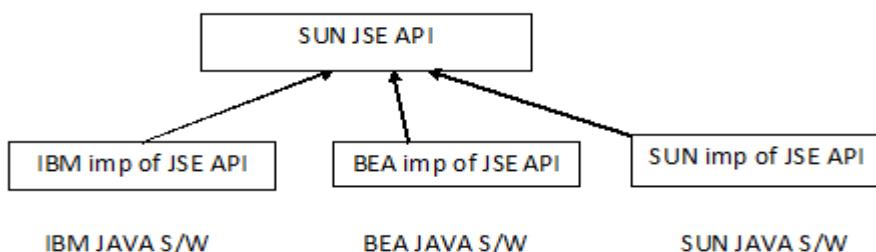
Any body can use the public API. Proprietary API can be used by only that company.

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

In case of 'C' language, API document contains set of predefined functions.

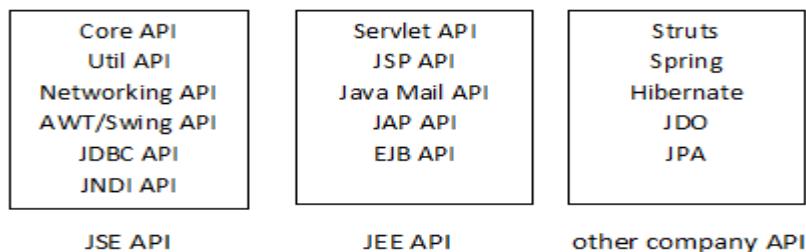
Once if the API is released any body can use the API.

The meaning of using API is providing the implementation.



Sun micro system has released JSE API. This is a public API released by sun micro system.

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



JDBC is an API released by sun micro system.

JDBC means for java data base connectivity.

We can store the data into multiple places they are

1. Files
2. Database server

They are so many disadvantages to store the data into file system. They are

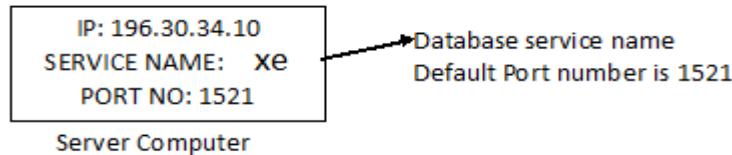
1. File system does not provide security.
2. There is limit on the size of the files.
3. If we store the data into file the redundant data will be stored.
4. We can resolve this entire problem by using data base servers only.
5. JDBC API is used if a java application want interact with database server.
6. There are so many database servers are available in the market. Some of them are Oracle, MYSQL, SAP, IBMDB2, POINT database server and etc.
7. When ever we purchase any database server we get two different software's they are Database Server Program(S/W)
 Database Client Program(S/W)

The database server s/w will be installed in server computer.

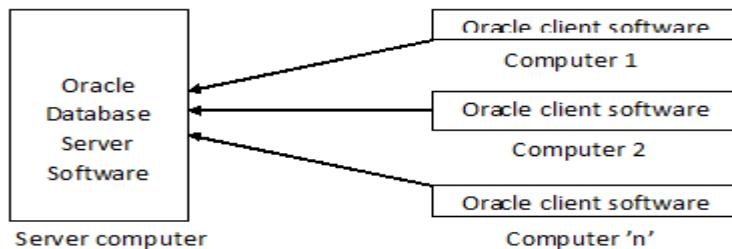
It is the responsibility of database or network administrator to install server software in server computer.

Every server computer is connected to a network. We can quickly identify a server computer by using the IP Address of the computer. If we want to develop any program to communicate with they are server with database three information's. They are

- 1) IP Address
- 2) Service Name
- 3) Port Number



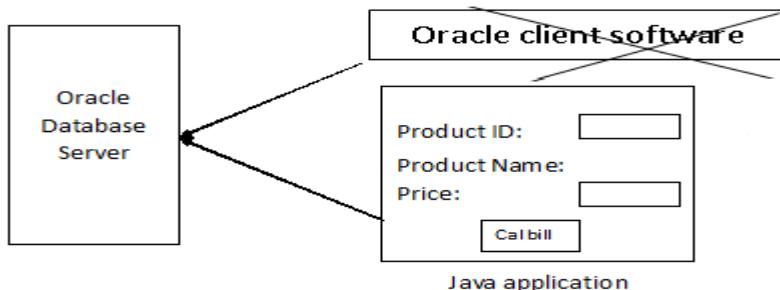
The database client software installed with all the developer computers.



Service name with nothing but the assigned to database server.

Port number is unique communications channel every programmer uses one port number.

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



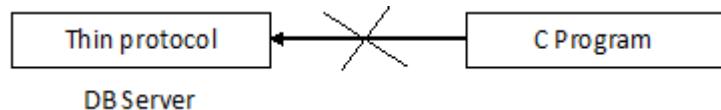
We using java application instead of oracle client program this is because by using oracle client program. If we want to request the server the client is suppose to type the SQL queries and sent the request to the server. But when we use a java application to send the request to the server, user no need to generate SQL queries. When the user interacts with GUI and the client click on the button the java application generate the queries and send to server.

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

Oracle database server is developing based on thin protocol.

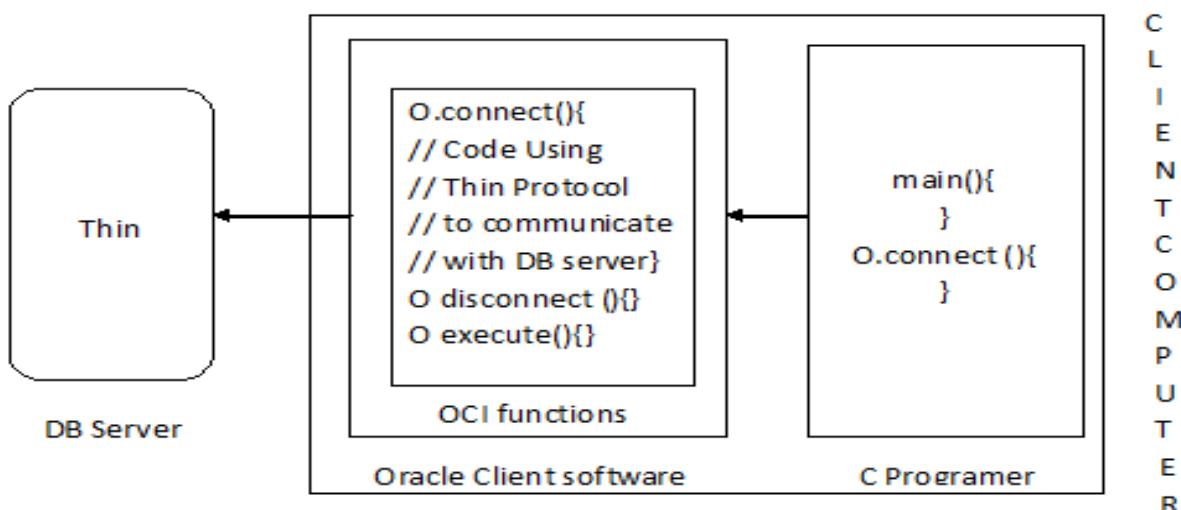
Thin protocol is proprietary to Oracle database server.

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



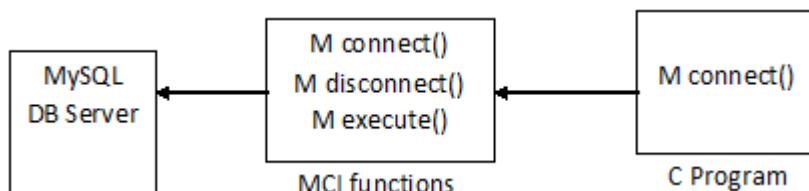
To resolve the above problem, Oracle guys are developed OCI functions (Oracle Call Interface), these functions are developed in 'C' language. The code inside the OCI functions interact with database server. Oracle guys are release OCI functions as part of client software.

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

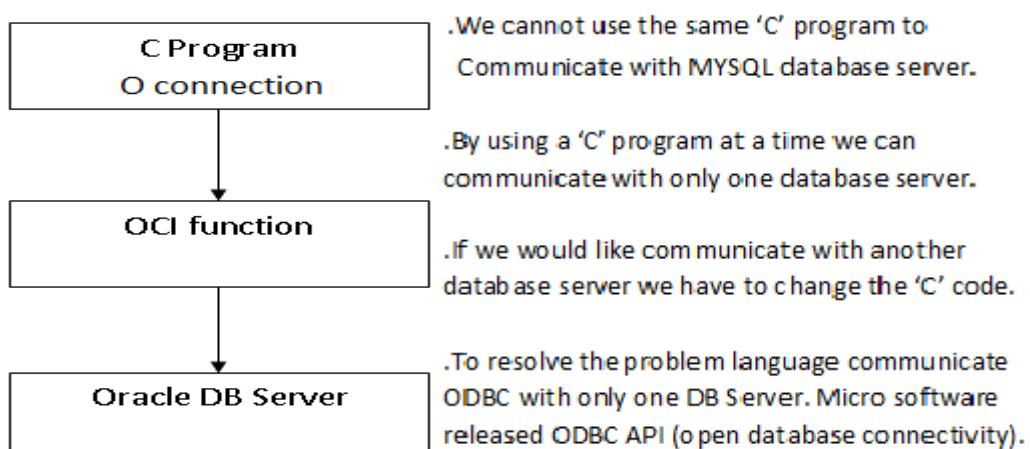


C
L
I
E
N
T
C
O
M
P
U
T
E
R

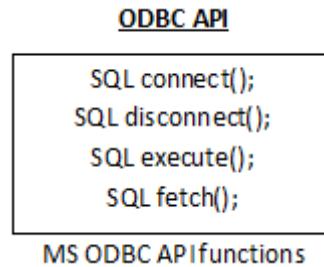
Similarly MYSQL DB Server releases MCI functions to interact with MYSQL DB Server.



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

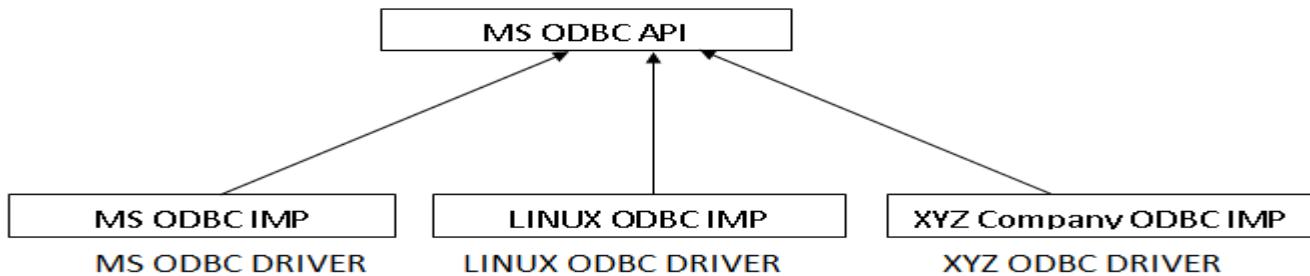


ODBC API contains set of 'C' language functions.

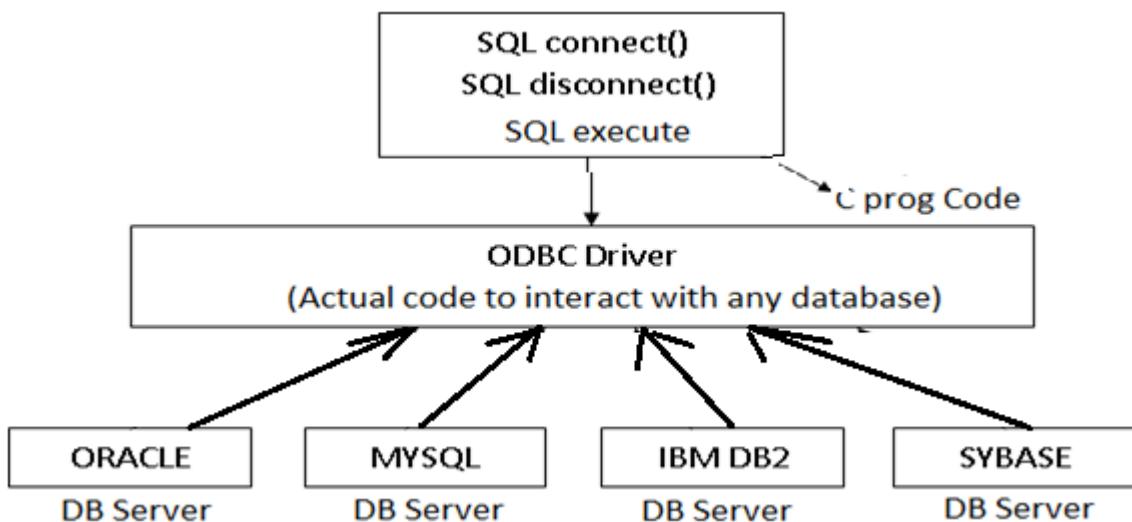


Once the API is released, any body can provide the implementation.

The implementation is called as ODBC software or ODBC Driver.



The following Architecture of 'C' program which uses ODBC API.



Sun Micro System released JDBC API to develop a java program to communicate with any database server without changing the java code.

The database server stores the data into files. These files are called as DBF files.



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

Oracle database server released in the form of two versions. They are:

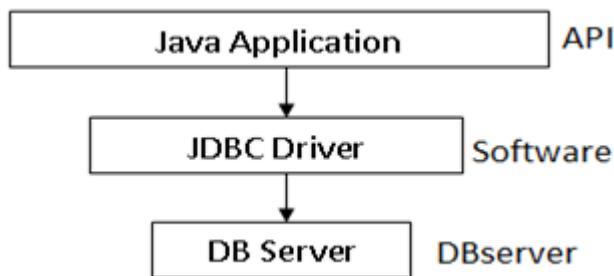
1. Enterprise edition (XE)
2. Express edition

As part of enterprise edition the service name port number of database server are fixed. They are

Service Name: XE

Port No: 1521

The following Architecture of JDBC Applications.



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

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

1. Java.sql
2. Javax.sql

The following are the most important of class and interface java.sql package.

*****java.sql interfaces:**

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

*****java.sql classes:**

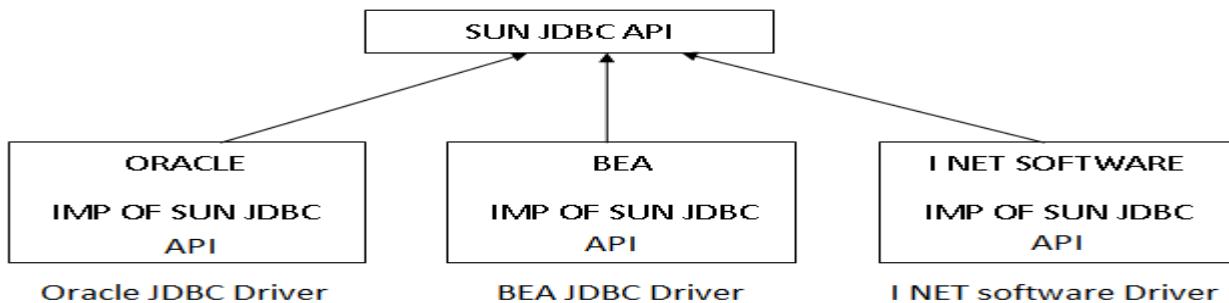
1. DriverManager
2. Types
3. Date

The most important (classes and) interfaces of javax.sql packages are:

****javax.sql interfaces:**

1. DataSource
2. RowSet

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



All the above companies release JDBC Driver software in the form of jar files.

We have to use this jar files to develop the java application to communicate with database server.

We have to following the procedure to develop a java application which communicate with DB server.

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

What is Driver class?

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

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

```
import java.sql.*;  
public class RegisterDriver{  
    public static void main(String[] args) throws SQLException{  
        Driver d = new oracle.jdbc.driver.OracleDriver();  
        DriverManager.registerDriver(d);  
        System.out.println("Driver is Registered");  
    }  
}
```

staticmethod driver object or driver class object

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

```
import java.sql.*;
public class DatabaseConnection{
    public static void main(String[] args) throws SQLException{
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:thin:@localhost:1521:xe","system","malli");
        System.out.println("got the connection:" + con.getClass());
    }
}
```

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

*Requirement:

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

```
import java.sql.*;
public class CreateTable{
    public static void main(String[] args) throws SQLException{
        //step1: Register the JDBC Driver
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        //step2: get the connection from the DB Server
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:thin:@localhost:1521:xe","system","malli");

        //step3: create the statement object
        Statement stmt = con.createStatement();
        //step4: execute the query
        stmt.executeUpdate(" create table emp(eno number(3), ename varchar2(15),
                           salary number(5))");

        //step5: close the connection
        Stmt.close();
    }
}
```

*in JDBC queries classified into two types they are:

1. Select query's
2. Non-select query

*what are non-select queries?

The query's which does not start with select keyword are called as non-select query's (insert, update, delete, create, drop and etc).

*what are the select queries?

The query's which starts with select keyword is called as select query.

To execute the non-select query's use a method execute updates.

Syntax: `int executeUpdate(query);`

To execute the select query we use a method execute query.

Syntax: `ResultSet executeQuery(query);`

In a project we create the tables any once before the project starts on the created tables are performing CURD operations.

C-Create	U-Update
R-Retrieve	D-Delete

***Requirement:** Develop a java program to create to insert a record into DBS to emp table. We should be able to insert a record into emp table. The values are empno 1, ename Raju and salary 1000.

```
import java.sql.*;  
public class InsertTable{  
public static void main(String args[])throws SQLException{  
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
Connection con =  
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");  
Statement stmt = con.createStatement();  
stmt.executeUpdate("insert into emp values(106,'Ravi',18000)");  
}  
}
```

Execute update method returns an integer value. This integer value indicates the number of records affected by the query in the database server. The memory of

effected is because of SQL query how many number of records are update, delete and inserted the data.

Ex: int no = stmt.executeUpdate("delete from emp where eno = 101");
System.out.println(no);
set CLASSPATH=ojdbc14.jar;:

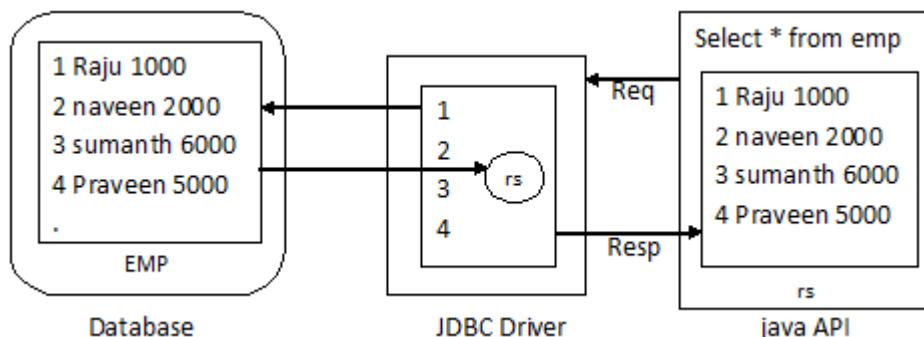
Requirement:

Develop the java application to retrieve all the records from emp table and display to the client.

When we execute the select query by using execute query method it's a returning ResultSet object by using System.out.println it is display object with has code it's not display the data available in ResultSet object.

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

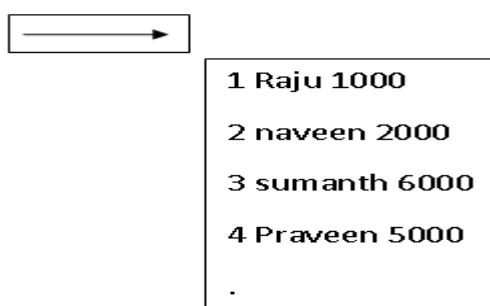
When we send the query to JDBC driver. The send the query to data base server. DBS is executed the query and return the records to JDBC driver. It is the responsibility of JDBC driver to convert data into ResultSet object (rs) and this ResultSet object is return to java application.



When even we got the result set object(rs). It is associated with a ResultSet pointer.

Initially these ResultSet pointer points to invisible record before first row.

ResultSet pointer



Get row method is used to find where the ResultSet pointer is pointing to.

Syn: **int getRow()**

To move the ResultSet pointer we use a method **next()**.

Syn: **boolean next()**

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

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

Datatype	java datatype	method names
Number	int	getInt("column name")
Varchar	String	getString("column name")
Double	double	getDouble("column name")
Number(10,2)	float	getFloat("column name")
Date	date	getDate("column name")

```
import java.sql.*;
public class RetriveRecords{
public static void main(String[] args)throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
while(rs.next()){
System.out.println(rs.getInt("eno"));
System.out.println(rs.getString("ename"));
System.out.println(rs.getInt("sal"));
}
}
```

In the JDBC API when ever an Exception is occur it will throw **java.sql.SQLException**.

As part of the JDBC program we are trying to retrieve column which is not available a ResultSet object. In this scenario java application throws SQLException with **invalid column name as the message**.

If the ResultSet pointer is pointing to an invisible record after the last row and if we are trying to get the values are get **exhausted ResultSet Exception**.

When we got the ResultSet object and without calling next() method we get an Exception **ResultSet next was not call**.

Most of the projects we try to select the records based on some criteria.

To filter the records based on the criteria as part of the select query we must use where clause. Retrieve all the records from emp table whose salary is > 3000.

Ex: `ResultSet rs = stmt.executeQuery("select * from emp where salary > 3000");`

*what is projection?

Display selected columns is called as projection.

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

Ex: `ResultSet rs = stmt.executeQuery("select eno,salary from emp");`

As part of the ResultSet interface we have over loaded methods.

Ex: `getxxx(columnName);`
 `getxxx(columnIndex);`

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

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

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

When even we use **DriverManager.getConnection** we get a physical connection between java.API and database server. It always recommended to close the connection other programs cannot use the connections of DBS.

To close the connection we use a method `con.close();` when we close the connection all the resources will be immediately released.

***Assignment:**

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

```
// using with do...while loop  
import java.sql.*;  
public class DoWhile{  
public static void main(String[] args)throws SQLException{  
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
Connection con=  
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");
```

```

Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
if(rs.next()){
do{
System.out.println(rs.getInt("eno"));
System.out.println(rs.getString("ename"));
System.out.println(rs.getFloat("sal"));
}while(rs.next());
}
else{
System.out.println("There is no Records");
}
}

import java.sql.*;
public class IfDoWhile{          //using with if condition and do...while
public static void main(String[] agrs)throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con=
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
if(rs.next()){
for(; rs.next() ;){
System.out.println(rs.getInt("eno"));
System.out.println(rs.getString("ename"));
System.out.println(rs.getFloat("sal"));
}
}
else{
System.out.println("There are no Records");
}
}
}

```

3. Try to create the table by using executeUpdate and check the return type of executeUpdate.

```
int no = executeUpdate("Query");
```

4. try to execute selectQuery by using executeUpdate and check the return value.

```
ResultSet rs = executeQuery("Query");
```

***PreparedStatement:**

PreparedStatement is also used to send the queries to DBS.

PreparedStatement improves the performance of java application when compared with statement object.

By using PreparedStatement also we can perform CURD operations.

C-create a record

U-update a record

R-retrieve records

D-delete a record

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

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

4. Supply the values to positional parameters by using setxxx() methods.
5. Execute the queries by calling executeUpdate() or executeQuery()
6. Close the connection

***Requirement:**

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

```
import java.sql.*;  
public class PreparedInsertRecord{  
public static void main(String[] args) throws SQLException{  
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
Connection con=  
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","malli");  
PreparedStatement pstmt = con.prepareStatement("insert into emp values(?, ?, ?)");  
pstmt.setInt(1,102);  
pstmt.setString(2,"Raju");  
pstmt.setDouble(3,5000);  
int no = pstmt.executeUpdate();
```

```

System.out.println(no);
con.close();
}
}

import java.sql.*;
public class PreparedStatementUpdate{
public static void main(String[] args)throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con= DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesavcherry");
PreparedStatement pstmt = con.prepareStatement("update student555 set sno = ?
where marks = ?");
pstmt.setInt(1,102);
pstmt.setFloat(2,99);
int no = pstmt.executeUpdate();
System.out.println(no);
con.close();
}
}

```

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

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

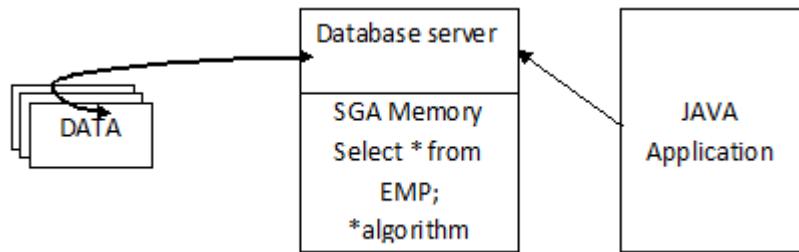
When we sent the query second time with a small changes in it , database server treats the query as different query and process all the steps.

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

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

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

- It execute the algorithm and get the data from database files and sent to client.



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

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

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

SQL>	variable Veno number; print Veno;
------	--------------------------------------

Ex: select * from emp where eno :veno := 5;
variable veno number;
exec := veno := 2;
insert into emp values(:veno, :vename, :vsalary);

*How to use bind variables in Oracle?

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

Syntax: variable variablename datatype

Ex: variable vsalary number;

- To **assign the value to variable** we use the following command.

Syntax: exec :variablename := value

Ex: exec :veno := 2;

- You **see the value** of variable we use performance a command.

Syntax: print variablename

Ex: print veno

- The following is an example of bind variable query to **insert the records**.

Ex: Insert into emp values(:veno, :vename, :vsalary);

- The following is an example of **select query from bind variable**.

Ex: **Select * from emp where eno =:veno**

6. When even the JDBC driver is ‘n’ commands prepared statement. The JDBC driver commands prepared statement into bind variables.
7. When even we call the **setter methods JDBC driver supply the values to bind variables.**
- 8. Prepared statement will not improve the performance when the query goes to database server for the first time. This is because first time database server has to execute all the steps.**
9. In a project we should never use Scott and tiger we always use our user name to login database server by using system user.
10. We have to use the following command to create a new user in database server.
 - a. Create user kesav identified by kesav cherry;
 - b. Grant connect, resource to kesav;

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

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

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

```
import java.io.*;  
import java.sql.*;  
public class PreparedBufferedReader{  
    public static void main(String arga[])throws IOException,SQLException{  
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
        System.out.println("Enter employee number:");  
        String eno = br.readLine();  
        System.out.println("Enter employee name:");  
        String name = br.readLine();  
        System.out.println("Enter employee salary:");  
        String salary = br.readLine();  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

```

Connection con= DriverManager.getConnection
        ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesavcherry");
String query = "insert into emp values(?, ?, ?)";
System.out.println(query);
PreparedStatement pstmt = con.prepareStatement(query);
pstmt.setString(1,eno);
pstmt.setString(2,name);
pstmt.setString(3,salary);
pstmt.executeUpdate();
    }
}

import java.util.*;
import java.sql.*;
public class Statementsanner{
public static void main(String[] args){
Scanner s = new Scanner(System.in);
System.out.println("eno");
int eno = s.nextInt();
System.out.println("name");
String name = s.next();
System.out.println("salary");
double salary = s.nextDouble();
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con= DriverManager.getConnection
        ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesavcherry");
Statement stmt = con.createStatement();
String query = "insert into emp values("+eno+',' "+name+" ','"+salary+"')";
System.out.println(query);
stmt.executeUpdate(query);
    }
}

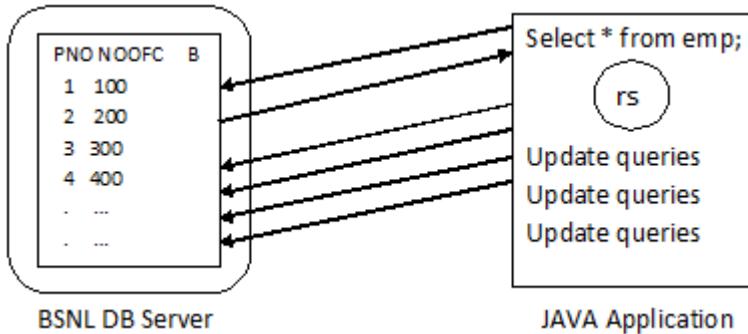
```

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

Callable statement:

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

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



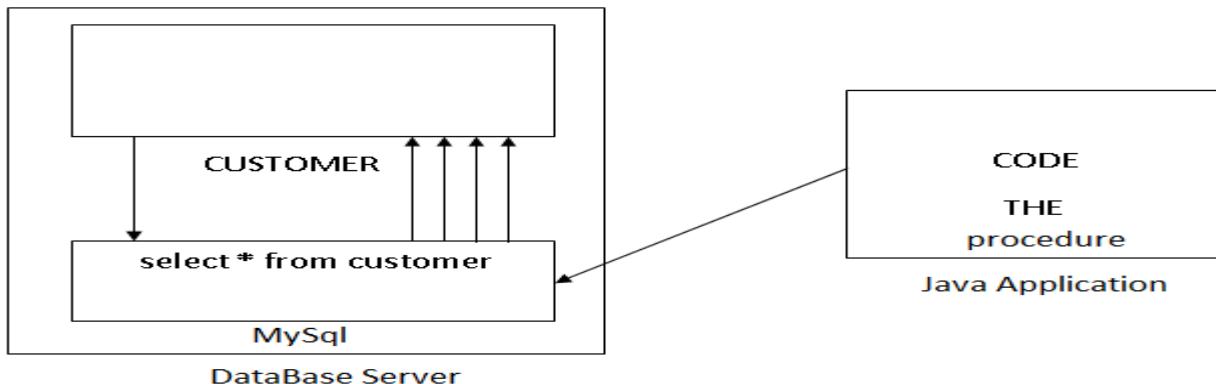
```

import java.sql.*;
public class BSNLbill{
public static void main(String[] args)throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesavcherry");
String query = "Select * from BSNLcustomer";
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery(query);
while(rs.next()){
int noofc = rs.getInt("noofc");
double bamount = noofc * .10;
int phoneno = rs.getInt("phoneno");
String UQuery = "update BSNLcustomer set bamount = ? where phoneno = ?";
//System.out.println(UQuery);
PreparedStatement pstmt = con.prepareStatement(UQuery);
pstmt.setDouble(1,bamount);
pstmt.setInt(2,phoneno);
pstmt.executeUpdate();
}
}
}
    
```

The disadvantage above approach we are in tract with database server from multiple times through networks. Because of this we get the performance issue we can resolve this by using processors.

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

When even any body call processor database server run's it processor improves performance of project.



We can use three types of procedures.

1. A procedure which doesn't take any type of parameters.
2. A procedure which takes input parameter.
3. A procedure input parameter as well as output parameter.

*How to create procedure in database?

create or replace procedure MyProc

as

begin

insert into emp values(2,'naveen',2000);

end MyProc;

/

*To execute the Procedure from the client we use a command exec MyProc;

The following procedure which takes input parameter.

1 create or replace procedure MyProc(veno IN number,

2 vname IN varchar2, vsalary in number)

3 as

4 begin

5 insert into emp values(veno, vname, vsalary);

6 end MyProc;

7 /

*To run the above procedure we use the following command exec MyProc(1,'abc',2345);

*A procedure which take input and output parameters.

create or replace procedure addition(no1 in number,

no2 in number, result out number)

as

begin

```
result := no1 + no2;
end addition;
/
exec addition;
exec addition(10,20, :result);
print result;
```

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

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

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

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

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

```
public class CallProcedure{
public static void main(String[] args) throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
CallableStatement cstmt = con.prepareCall("{call MyProc}");
cstmt.close();
}
}
```

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

SQL>

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

```
import java.util.*;
import java.sql.*;
public class CallProcedure{
public static void main(String[] args) throws SQLException{
Scanner s = new Scanner(System.in);
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

```

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

```

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

create or replace procedure addition(Vno1 in number,
Vno2 in number, Vresult out number)

as

begin

result := Vno1 + Vno2;

end addition;

/

```

import java.sql.*;
public class CallableStatement{
public static void main(String[] args)throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
CallableStatement cstmt = con.prepareCall("{call addition(?, ?, ?)}");
cstmt.setInt(1,11);
cstmt.setInt(2,22);
cstmt.registerOutParameter(3,Types.INTEGER);
cstmt.execute();
int result = cstmt.getInt(3);
System.out.println(result);
}
}

```

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

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

In oracle we have the following aggregate functions they are

1. count()
2. min()
3. max()
4. sum()
5. avg() and etc.

To call function in oracle we use Queries for example

Select sum(sal) from emp;

Select max(sal) from emp;

Select count(*) from emp;

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

```
import java.sql.*;  
public class FindRecords{  
    public static void main(String[] main)throws SQLException{  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
        Connection con = DriverManager.getConnection  
            ("jdbc:oracle:thin:@localhost","kesav","kesav cherry");  
        Statement stmt = con.createStatement();  
        ResultSet rs = stmt.executeQuery("select count(*) count from emp");  
        if(rs.next()){  
            System.out.println(rs.getInt("count"));  
        }  
    }  
}
```

Select * from user_source;

Show errors;

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

as

result number;

begin

```
result := Vno1 + Vno2;  
return result;  
end addition;  
/
```

Approach1: To call the above functions use a query to get the values.

SQL> select addition(10,20) from dual;

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

SQL> select addition(10,20) sum from dual;

Ex: import java.sql.*;

```
public class FindRecords{  
    public static void main(String[] args)throws SQLException{  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
        Connection con = DriverManager.getConnection  
            ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");  
        CallableStatement cstmt = con.prepareCall("{? = call addition(?,?)}");  
        cstmt.registerOutParameter(1,Types.INTEGER);  
        cstmt.setInt(2,23);  
        cstmt.setInt(3,33);  
        cstmt.execute();  
        int sum = cstmt.getInt(1);  
        System.out.println(sum);  
    }  
}
```

***Types of ResultSets:** They are two types of ResultSets in java. They are

1. Forwardonly ResultSet
2. Bi-Directional ResultSet

***What is forwardonly ResultSet?**

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

***What is Bi-Directional ResultSet?**

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

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

Syntax: To create Bi-Directional ResultSet for statement object.

```
create statement(resultSet TYPE, resultSet CON CURRENCY);
```

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

1. TYPE_FORWARD_ONLY —————→ By Default
2. TYPE_SCROLL_SENSITIVE
3. TYPE_SCROLL_INSENSITIVE

For ResultSet Con Currency we can supply any of the following two values.

1. CONCUR_READ_ONLY —————→ By Default
2. CONCUR_UPDATABLE

*What is Sensitive ResultSet?

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

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

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

Ex:

```
import java.sql.*;  
public class ResultSetTypeStatement{  
    public static void main(String[] args) throws SQLException{  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
        Connection con = DriverManager.getConnection  
            ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");  
        Statement stmt = con.createStatement  
            (ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_READ_ONLY);  
        ResultSet rs = stmt.executeQuery("select * from emp");  
        rs.next();  
        rs.next();  
        System.out.println(rs.getRow());  
        rs.previous();  
        System.out.println(rs.getRow());  
        rs.previous();  
        System.out.println(rs.getRow());  
        con.close();  
    }  
}
```

}

When we call the previous() method the ResultSet pointer is placed to old record.

Absolute(): This method is used to move the ResultSet pointer to a specific Record/Row.

Ex: rs.absolute(5);

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

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

Ex: PreparedStatement pstmt = con.prepareStatement("select * from emp",
 .ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_READ_ONLY);

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

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

```
import java.sql.*;  
public class ResultSetTypeReadOnly{  
    public static void main(String[] args) throws Exception{  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
        Connection con = DriverManager.getConnection  
            ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");  
        Statement stmt = con.createStatement  
            (ResultSet.TYPE_SCROLL_SENSITIVE,ResultSet.CONCUR_READ_ONLY);  
        ResultSet rs = stmt.executeQuery("select eno, ename, sal from emp");  
        while(rs.next()){  
            //System.out.println("press any key to get the next result");  
            //System.in.read();  
            //System.in.read();  
            rs.refreshRow();  
            System.out.println(rs.getString(1));  
            System.out.println(rs.getString(2));  
            System.out.println(rs.getString(3));  
        }  
    }  
}
```

Whenever java program has encountered a `refreshRow()` it will check the data in database server and the data in `ResultSet` object same or not. If they are not same `refreshRow()` will update the data in `ResultSet` object.

refreshRow() method work only from sensitive ResultSet.

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

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

Ex:

```
import java.sql.*;  
public class Absolute{  
    public static void main(String[] args) throws SQLException{  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
        Connection con = DriverManager.getConnection  
            ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");  
        Statement stmt = con.createStatement  
            (ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);  
        ResultSet rs = stmt.executeQuery("select eno, ename, sal from emp");  
        rs.absolute(3);  
        rs.updateDouble("sal",22000);  
        rs.updateRow();  
    }  
  
    import java.sql.*;  
    public class MoveToInsertRow{  
        public static void main(String[] args) throws SQLException{  
            DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
            Connection con = DriverManager.getConnection  
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");  
            Statement stmt = con.createStatement  
                (ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);  
            ResultSet rs = stmt.executeQuery("select eno,ename,sal from emp");  
            rs.moveToInsertRow();  
            rs.updateInt(1,123);  
            rs.updateString(2,"veera");  
            rs.updateDouble(3,19000);  
            rs.insertRow();  
        }  
    }
```

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

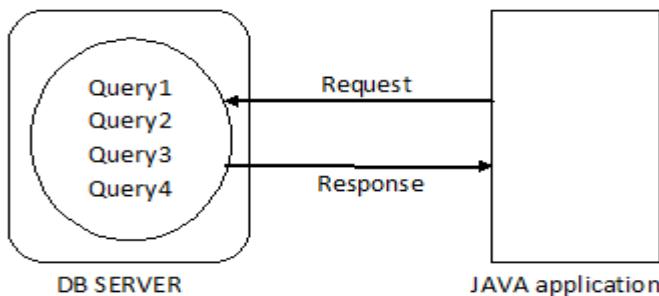
To achieve the above requirement we can a method delete Row.

```
import java.sql.*;
public class AbsoulteDeleteRow{
    public static void main(String[] args) throws SQLException{
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection con = DriverManager.getConnection
            ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
        Statement stmt = con.createStatement
            (ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);
        ResultSet rs = stmt.executeQuery("select eno,ename,sal from emp");
        rs.absolute(9);
        rs.deleteRow();
    }
}
```

Batch Updates: (improve the performance)

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

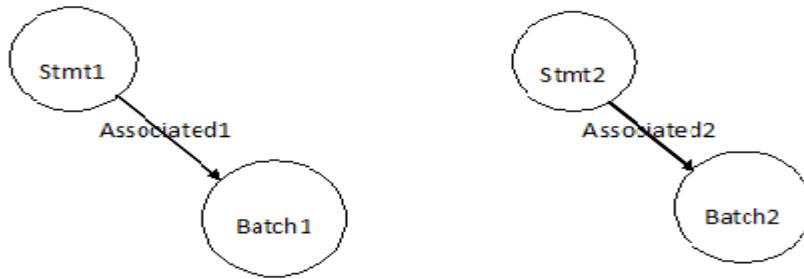
If we use Batch updates we can write Business logic in java applications.



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

Ex: Statement stmt1 = con.createStatement();

Statement stmt2 = con.createStatement();



Requirement:

*Develop a java application to insert three records by using Batch updates?

```

import java.sql.*;
public class InsertBatchRecords{
public static void main(String[] args)throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
Statement stmt = con.createStatement();
stmt.addBatch("insert into emp values(122,'niranjan',16666)");
stmt.addBatch("insert into emp values(124,'sravan',11143)");
stmt.addBatch("insert into emp values(125,'mrudhu',14371)");
stmt.executeBatch();
}
}

```

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

Syntax of execute:

int a[] executeBatch();

Ex: int a[] = stmt.executeBatch();

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

Ex: for(int i = 0;i<a.length;i++){
System.out.println(a[i]);
}

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

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

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

```
import java.sql.*;
public class PreparedBatchInsert{
public static void main(String[] args) throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
PreparedStatement pstmt = con.prepareStatement("insert into emp values(?, ?, ?)");
pstmt.setInt(1,133);
pstmt.setString(2, "suneel");
pstmt.setDouble(3,15500);
pstmt.addBatch();
pstmt.setInt(1,144);
pstmt.setString(2, "sudheer");
pstmt.setDouble(3,9876);
pstmt.addBatch();
pstmt.executeBatch();
}
}

import java.sql.*;
public class PreparedBsnlBill{
public static void main(String[] args) throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe", "kesav", "kesav cherry");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from BSNLCUSTOMER");
PreparedStatement pstmt = con.prepareStatement
                ("update bsnlcustomer set bamount = ? where phoneno = ?");
while(rs.next()){
int nofc = rs.getInt("nofc");
int phoneno = rs.getInt("phoneno");
double bamount = nofc * 0.10;
pstmt.setDouble(1,bamount);
pstmt.setInt(2,phoneno);
pstmt.addBatch();
}
```

```

        }
        pstmt.executeBatch();
    }
}

```

Without JDBC driver we cannot develop java application.

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

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

Ex: public class RetriveRecords{
 public static void main(String[] args){
 class.forName("oracle.jdbc.driver.OracleDriver");
 Connection con = DriverManager.getConnection
 ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav chari");
 }
}

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

Ex: public class oracleDriver implements Driver{
 //provides the implement to the all 6 methods
 static{
 OracleDriver o = new oracleDriver();
 DriverManager.registerDriver(o);
 }
}

Sun micro has released different versions of JDBC specification.

Ex: JDBC 3.0 and JDBC 4.0

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

Ex: Incase of JDBC 3.0 specification oracle ways are chooseojdbc14.jar in case of JDBC 4.0 implementation they chooseojdbc6.jar.

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

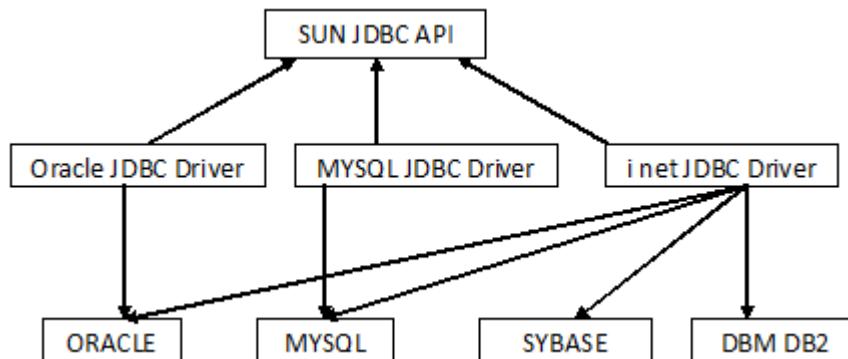
As part of JDBC4.0 specification sun micro system has added a feature to register the driver automatically (DriverManager.getConnection).

We can find the specification implementation version as part of META_INF folders.

When we work with JDBC4.0 to register JDBC driver automatically we need to make sure that we set the CLASSPATH to ojdbc6.jar.

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

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



*uses the enterprise edition;

USERNAME: root

PASSWORD: kesav

*procedure to work with mysql database server.

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

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

Ex: create database database mydb.

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

*Develop a java application to insert a record into emp table of mysql database server?

```
import java.sql.*;  
public class MysqlInsertRecords{  
    public static void main(String[] args) throws Exception {  
        Class.forName("com.mysql.jdbc.Driver");  
        Connection con = DriverManager.getConnection  
            ("jdbc:mysql://localhost:3306/mydb","root","kesav");  
        Statement stmt = con.createStatement();  
        int no = stmt.executeUpdate("insert into emp values(999,'Mohan',60000);  
        System.out.println(no);
```

```
    }  
}
```

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

*The following an example of java application which can be used to interact with any database server?

```
import java.sql.*;  
  
public class AnyDatabaseServerInsertRecord{  
    public static void main(String[] args) throws SQLException, ClassNotFoundException{  
        String drv = System.getProperty("drv");  
        String url = System.getProperty("url");  
        String uname = System.getProperty("uname");  
        String pwd = System.getProperty("pwd");  
        Class.forName(drv);  
        Connection con = DriverManager.getConnection(url,uname,pwd);  
        Statement stmt = con.createStatement();  
        int no = stmt.executeUpdate("insert into emp values(888,'RajuRani',143);  
        System.out.println(no);  
    }  
}
```

imp: By removing the hard coding and if we would like to use DriverManager.registerDriver we can use following approach.

```
public class HardcodingInsertRecords{  
    public static void main(String[] args) throws Exception{  
        String drv = System.getProperty("drv");  
        Class c = Class.forName(drv);  
        Object o = c.newInstance();  
        Driver d = (Driver)o;  
        DriverManager.registerDriver(d);  
        Connection con = DriverManager.getConnection(".....");  
        Statement stmt = con.createStatement();  
        int no = stmt.executeUpdate("insert into emp values(1,'Raju',1000)");  
        System.out.println(no);  
    }  
}
```

```
    }  
}
```

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

Transactions: (performing)

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

*The following is an example of user defined transactions?

```
import java.sql.*;  
public class AutoCommitInsertRecords{  
    public static void main(String[] args) throws SQLException{  
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());  
        Connection con= DriverManager.getConnection("url","uname","pwd");
```

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

ENO	<input type="text"/>				
ENAME	<input type="text"/>				
SALARY	<input type="text"/>				
ADDRESS:					
Street	<input type="text"/>	City	<input type="text"/>	State	<input type="text"/>
Street	<input type="text"/>	City	<input type="text"/>	State	<input type="text"/>
Street	<input type="text"/>	City	<input type="text"/>	State	<input type="text"/>

[Click here to add new Address](#)

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

To achieve the above requirement we have two designs.

Design1: In this design we create only table.

ENO	NAME	SALARY	STREET	CITY	STATE
1	Raju	10000	Ameetpet	HYD	AP
2	Raju	10000	Ameerpet	HYD	AP
--	--	-----	-----	---	--
--	--	-----	-----	---	--

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

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

ENO	NAME	SALARY	ENO	STREET	CITY	STATE
1	Raju	10000	1	Ameerpet	HYD	AP
			1	SRnagar	HYD	AP

EMP table

ADDRESS table

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

```
import java.sql.*;  
import java.util.*;  
public class InsertTwoTableRecords{  
    public static void main(String[] args) throws SQLException{  
        Scanner s = new Scanner(System.in);
```

```

System.out.println("Enter the employee number");
int eno = s.nextInt();
System.out.println("Enter the employee ename");
String ename = s.next();
System.out.println("Enter the employee salary");
double salary = s.nextDouble();
System.out.println("Enter the employee street");
String street = s.next();
System.out.println("Enter the employee city");
String city = s.next();
System.out.println("Enter the employee state");
String state = s.next();
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
con.setAutoCommit(false);
PreparedStatement pstmt1 = con.prepareStatement("insert into emp2 values(?, ?, ?)");
pstmt1.setInt(1, eno);
pstmt1.setString(2, ename);
pstmt1.setDouble(3, salary);
PreparedStatement pstmt2 = con.prepareStatement
                ("insert into address values(?, ?, ?, ?)");
pstmt2.setInt(1, eno);
pstmt2.setString(2, street);
pstmt2.setString(3, city);
pstmt2.setString(4, state);
pstmt1.executeUpdate();
pstmt2.executeUpdate();
con.commit();
}
}

```

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

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

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

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

```
import java.sql.*;
import java.util.*;

public class DBConnectionTwoTables{
    public static void main(String[] args) throws Exception{
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the employee number");
        int eno = s.nextInt();
        System.out.println("Enter the employee ename");
        String ename = s.next();
        System.out.println("Enter the employee salary");
        double salary = s.nextDouble();
        System.out.println("Enter the employee street");
        String street = s.next();
        System.out.println("Enter the employee city");
        String city = s.next();
        System.out.println("Enter the employee state");
        String state = s.next();
        Connection con = null;
        PreparedStatement pstmt1 = null;
        PreparedStatement pstmt2 = null;
        ResultSet rs = null;
        con.setAutoCommit(false);
        try{
            DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
            con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","system","malli");
            con.setAutoCommit(false);
            pstmt1 = con.prepareStatement("insert into emp2 values(?, ?, ?)");
            pstmt1.setInt(1, eno);
            pstmt1.setString(2, ename);
            pstmt1.setDouble(3, salary);
            pstmt2 = con.prepareStatement("insert into address values(?, ?, ?, ?)");
        }
```

```
pstmt2.setInt(1,eno);
pstmt2.setString(2,street);
pstmt2.setString(3,city);
pstmt2.setString(4,state);
pstmt1.executeUpdate();
pstmt2.executeUpdate();

        }catch(SQLException se){
con.rollback();
System.out.println("some code is missing");
}finally{
con.close();
pstmt1.close();
pstmt2.close();
rs.close();
        }
    }
}
```

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

```
import java.sql.*;
public class DBServerConnectType4C{
public static void main(String[] args)throws SQLException{
Connection con = null;
Statement stmt = null;
ResultSet rs = null;
try{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
con = DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe","system","malli");
stmt = con.createStatement();
rs = stmt.executeQuery("select * from emp");
System.out.println(con);
System.out.println(stmt);
System.out.println(rs);
}catch(SQLException s){
}finally{
```

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

*connection is an interface:

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

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

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

How to achieve polymorphism by using JDBC?

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

*MetaData:

Data about data is called as MetaData.

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

1. ResultSetMetaData
2. DatabaseMetaData
3. ParameterMetaData

*ResultSetMetaData:

ResultSetMetaData is an object which gives more information about ResultSet object. By using ResultSetMetaData object we can find the number of column is available in ResultSet, the names of the columns, the Data types of the columns.

Ex: ResultSetMetaData rsmd = rs.getMetaData();

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

*ParameterMetaData:

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

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

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

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

```
ResultSetMetaData rsmd = rs.getMetaData();
System.out.println(rsmd.getColumnCount());
System.out.println(rsmd.getColumnName(2));
System.out.println(rsmd.getColumnTypeName(2));
System.out.println(rsmd.isSearchable(2));
System.out.println(rsmd.getColumnType(1));
```

*Write a sample program to use ResultSetMetaData?

```
import java.sql.*;
public class RSMDConnection{
public static void main(String[] args) throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
ResultSetMetaData rsmd = rs.getMetaData();
System.out.println(rsmd.getColumnCount());
System.out.println(rsmd.getColumnName(2));
System.out.println(rsmd.getColumnTypeName(2));
System.out.println(rsmd.isSearchable(2));
System.out.println(rsmd.getColumnType(2));
    }
}
```

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

```
import java.sql.*;
public class RSMDTryCatch{
public static void main(String[] args) throws SQLException{
Connection con = null;
Statement stmt = null;
ResultSet rs = null;
```

```

try{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
stmt = con.createStatement();
rs = stmt.executeQuery("select * from emp");
ResultSetMetaData rsmd = rs.getMetaData();
System.out.println(rsmd.getColumnCount());
System.out.println(rsmd.getColumnName(2));
System.out.println(rsmd.getColumnTypeName(2));
System.out.println(rsmd.isSearchable(2));
System.out.println(rsmd.getColumnType(1));
}catch(SQLException s){
System.out.println("some problem in the above code");
}finally{
con.close();
stmt.close();
rs.close();
}
}
}

```

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

***DatabaseMetaData:**

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

```

import java.sql.*;
public class DatabaseMetaDataConnection{
public static void main(String[] args) throws SQLException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
DatabaseMetaData dbmd = con.getMetaData();

```

```
System.out.println(dbmd.getDatabaseMajorVersion());
System.out.println(dbmd.getDatabaseMinorVersion());
System.out.println(dbmd.getDatabaseProductName());
System.out.println(dbmd.getDriverMajorVersion());
System.out.println(dbmd.getDriverMinorVersion());
System.out.println(dbmd.getJDBCMajorVersion());
System.out.println(dbmd.getJDBCMinorVersion());
}
}
```

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

***ParameterMetaData:**

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

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

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

***Types of Drivers:**

They are 4 types of JDBC Drivers are available in the market.

TYPE 1: JDBC-ODBC Bridge Driver

TYPE 2: Native API Driver

TYPE 3: Network protocol Driver

TYPE 4: Pure java Driver/thin Driver

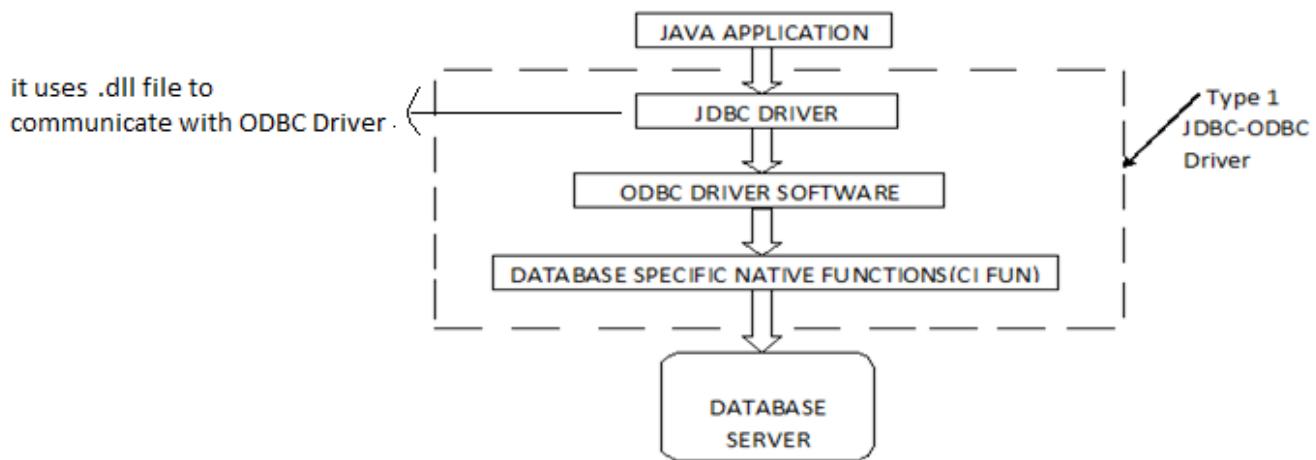
***JDBC-ODBC Bridge Driver:**

Sql * plus is a client S/W , It is developed in C lang , it internally uses OCI.dll file to communicate with DB server . This OCI.dll having all C lang functions to communicate with Oracle DB , some functions are OCILogon , OCILogoff Etc.

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

The following architecture of type1 JDBE-ODBC Driver.

JDBC Driver is Developed in JAVA , ODBC Driver is Developed in C language.



***Disadvantages of type 1 driver:**

1. If we would like work with type one JDBC driver. Developer/Client as to install software of database to get CI functions we have install ODBC Driver software and JDBC driver software and JDBC driver software. We have to configure JDBC driver to communicate with ODBC Driver.
2. Type1 Driver is platform dependent this is because CI functions are developed in 'C' language.
3. MS as released ODBC API to develop a 'C' program to interact with any database server. The internal code of the ODBC drivers uses OCI functions(native functions) to communicate with Database servers.
4. If the .DLL's are not available in ODBC driver software we have install them separately.
5. By default ODBC Driver available with windows operations system.
(Start → control panel → administrative tools → data source)
6. Type1 JDBC Driver is developed by only sun micro system. No body else has developed type1 JDBC Driver. Sun micro system has integrated type1 JDBC Driver with java software only. We no need to install type1 JDBC Driver saperatly.
7. When we observe the .DLL file of oracle we have found that internally it is use its ODBC API functions. The function survive SQL connect, SQL disconnect, SQL execute and etc. these functions internally communicate with OCI functions.

***Procedure to configure ODBC Driver:**

- Open ODBC data source administrative window.
- Choose system DSN tab and click on add button.
- From the list of available ODBC drivers choose the required driver (Oraclexe).

- From the driver configure window provide data source name, description, userid and click on “OK”.

*The following is an example of using type1 Database connection Driver?

```
import java.sql.*;
public class InsertRecords{
public static void main(String[] args){
DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());
Connection con = DriverManager.getConnection
                ("jdbc:odbc:myoracleds","kesav","kesavcherry");
Statement stmt = con.createStatement();
stmt.executeUpdate("insert into emp values(1,'abc',234)");
}
}
```

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

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

***Advantage of type1 Driver:**

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

***Requirement:**

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

MS Access is not a DB server , it is a File.

```
DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());
Connection con = DriverManager.getConnection("jdbc:odbc:mysqlds","kesav","kesav");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
```

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

When we are using excel sheet the sheet name as tread as table name. the first row is treated as columns names.

MS Excell is not a DB server , it is a File

```

DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());
Connection con = DriverManager.getConnection("jdbc:odbc:myds","","","");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from [sheet1$]");
while(rs.next()){
    System.out.println(rs.getString(1));
    System.out.println(rs.getString(2));
}

```

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

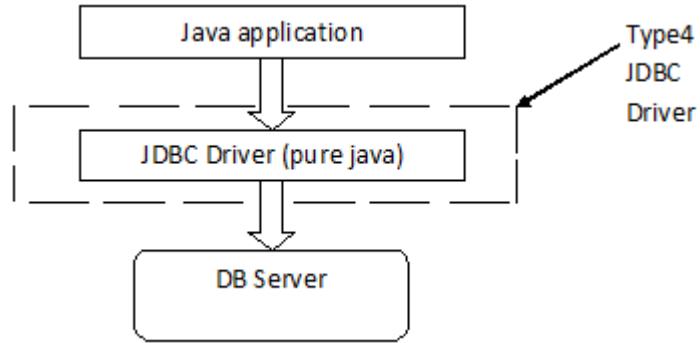
```

import java.sql.*;
public class RetriveRecords3{
public static void main(String[] args){
DriverManager.registerDriver(new sun.jdbc.odbc.jdbcOdbcDriver());
Connection excelcon = DriverManager.getConnection("jdbc:odbc:myds","","","");
Statement excelstmt = excelcon.createStatement();
ResultSet rs = excelstmt.executeQuery("select * from [emp$]");
Connection oraclecon = DriverManager.getConnection
                    ("jdbc:odbc:oracleds","kesav","kesav cherry");
PreparedStatement pstmt = oraclecon.prepareStatement
                    ("insert into emp values(?, ?, ?, ?)");
while(rs.next()){
    pstmt.setString(1,rs.getString(1));
    pstmt.setString(2,rs.getString(2));
    pstmt.setString(3,rs.getString(3));
    pstmt.addBatch();
}
pstmt.executeBatch();
}
}

```

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

The following is architecture of type4 JDBC Driver.



***Advantages of type4 driver:**

This Driver is development in pure java. Because of those we can say this Driver is platform independent.

This is fastest Driver in all the Drivers. This is because Driver directly communicates with database server.

We need to install any client software.

***Disadvantage of type4 driver:**

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

***JNI:**

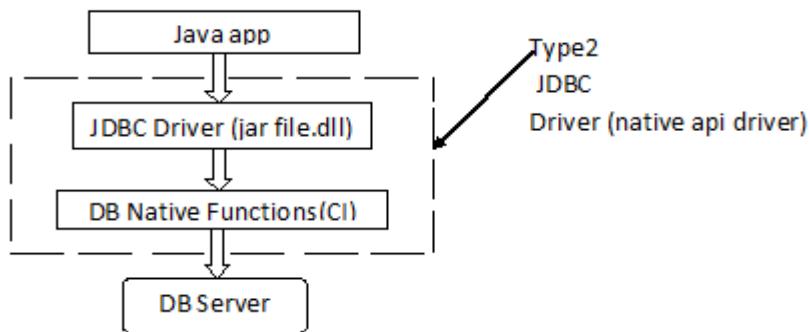
JNI stands for Java Native Interface, **JNI is an API which is used to develop a java program to call the 'C' language functions. JNI is Developed in C language.**

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

```

Ex: public class HelloWorld{
        public native void SayHello(){}
        static{
            System.loadLibrary("Hello"); // Hello is the function available in C lang
        }
        public static void main(String[] args){
            HelloWorld h = new HelloWorld()
            h.SayHello();
        }
    }
  
```

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



Oracle people have placed all the .DLL which is required to interact with OCI functions in the bin folder. As part of oracle10g the name of the DLL which uses OCI functions is OCIJDBC10.dll.

JDBC Driver (ojdbc14.jar) interact with ocijdbc10.dll , and ocijdbc.dll interact with oci.dll, because of JDBC Driver can't communicate with oci.dll file Directly.Because of JDBC Driver Developed in JAVA and oci.dll file is developed in C lang ,so to interact java code with c lang , we need an other c lang code as intermediate .

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

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

Ex:

```

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:oci:@localhost:1521:xe","kesav","kesav cherry");
PreparedStatement pstmt = con.prepareStatement("insert into emp2 values(?, ?, ?)");
pstmt.setInt(1,3);
pstmt.setString(2,"kkk");
pstmt.setDouble(3,66666);
pstmt.executeUpdate();
pstmt.executeBatch();
pstmt.addBatch();

```

***Disadvantages of Type2 Driver:**

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

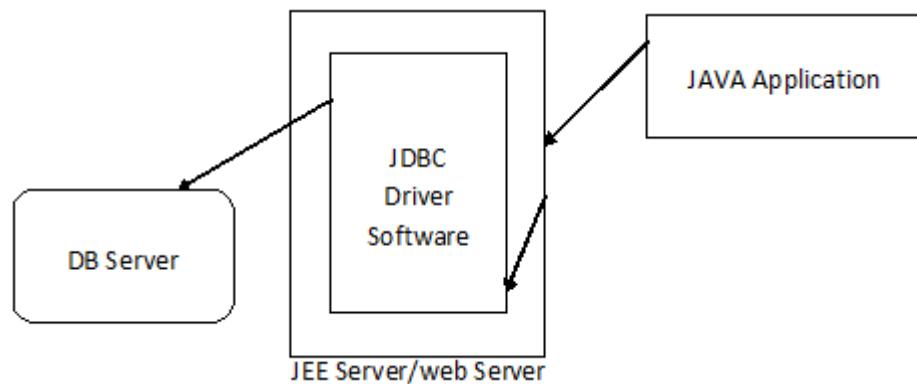
Like Type1 Driver it has to translate the java language calls to ‘C’ language calls.

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

Type2 Driver gives better performance when compared with Type1 Driver.

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

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



***Requirement:**

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

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

Create table

```

import java.sql.*;
import java.io.*;
public class StoreImage{
public static void main(String[] args)throws SQLException,FileNotFoundException{
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
PreparedStatement pstmt = con.prepareStatement("insert into empimg values(?, ?, ?)");
pstmt.setInt(1,4);
pstmt.setString(2,"anc");
File f = new File("Jellyfish.jpg");
FileInputStream fis = new FileInputStream(f);
pstmt.setBinaryStream(3,fis,(int)f.length());
pstmt.executeUpdate();
}
}
  
```

JDBC code To retrieve Image from DB server.

```

import java.sql.*;
import java.io.*;
  
```

```

public class RetrieveImage{
public static void main(String[] args){
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
Connection con = DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav cherry");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from empimg");
rs.next();
System.out.println(rs.getInt(1));
System.out.println(rs.getString(2));
byte imgarr[] = Rs.getBytes(3);
FileOutputStream fos = new FileOutputStream("one.jpg"); // target folder to place img
}
}

```

JDBC Code to Store Image and File into a DB Server.

```

SQL> create table files (eno number(5), image blob, file clob);

import java.io.*;
import java.sql.*;

public class StoreFile {
    public static void main(String[] args) {
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con=DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav");
            String query="insert into files values(?, ?, ?)";
            PreparedStatement ps=con.prepareStatement(query);
            System.out.println("The query is "+query);
            File f=new File("1.jpeg");
            FileInputStream fr=new FileInputStream(f);

            File f1=new File("aadhaar.pdf");
            FileReader fr1=new FileReader(f1);

            ps.setInt(1,101);
            ps.setBinaryStream(2,fr,(int)f.length());
            ps.setCharacterStream(3,fr1,(int)f1.length());
            int i=ps.executeUpdate();
        }
    }
}

```

```

        System.out.println(i+" records affected");
        con.close();
    }catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

JDBC Code to Retrieve Image and File from a DB Server.

```

import java.io.*;
import java.sql.*;
public class RetrieveFile {
public static void main(String[] args) {
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection
                ("jdbc:oracle:thin:@localhost:1521:xe","kesav","kesav");
PreparedStatement ps=con.prepareStatement("select * from files");
ResultSet rs=ps.executeQuery();
rs.next();//now on 1st row
System.out.println(rs.getInt(1));
byte[] imgs = rs.getBytes(2);
Reader r = rs.getCharacterStream(3);

FileOutputStream img = new FileOutputStream("E:/JavaProgs/JDBC/ab.jpeg");

FileWriter fw=new FileWriter("E:/JavaProgs/JDBC/eaadhar.pdf");
Img.write(imgs);
fw.close();
con.close();

System.out.println("success");
}catch (Exception e) {e.printStackTrace();  }
}
}

```

*Date:

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

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

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

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

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

Ex:

```
J Button.addActionListener(new ActionListener(){  
    public void actionPerformed(ActionEvent e){  
        try{  
            //JDBC code to store in DB Server  
        }catch(SQLException s){  
            s.printStackTrace();  
        }  
    }  
})
```

JNDI

JNDI: (Java Naming and Directory Interface)

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

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

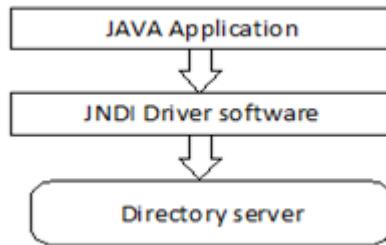
As part of **Database server we store the data in the form of relational records** as part of **Directory server we store the data in the form of objects.**

Difference between Database server and Directory servers:

- Directory servers give the best performance if we want to the store the data once and retrieve the data for multiple times.
- As part of Database servers we can store huge amount of data. Directory servers are not meant to store huge amount of data. They can store only small amount of data. (Yahoo, Google and etc...)
- To interact with database servers where using query language. To interact with directory server we have to use the predefined functions/methods.
- Directory servers can't store the data permanently. These servers are meant to store the data temporally. In most of the project we use both database servers and directory servers.

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

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



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

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

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

Ex:

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

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

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

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

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

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

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

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

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

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

Enter the username and password which act as on administrator.

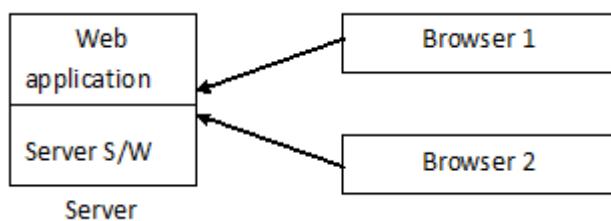
Select the development mode and available JDK. Choose the customization as NO and click on create BUTTON.

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

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

The advantage of this approach is we now no longer need to install the client software separately. We can access the client software by taking the help of BROWSER. To access oracle client software we use the following URL.

Ex: <http://localhost:8080/apex/>

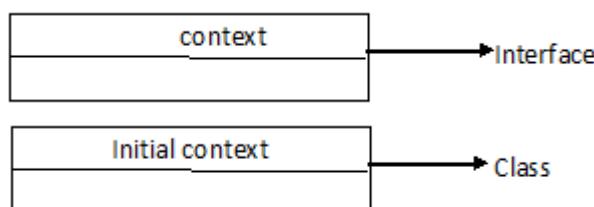


The following is URL which is used to access admin console of web logic server.

Ex: <http://localhost:7001/console/>

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

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



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

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

JDBC	JNDI
driver class	INITIAL_CONTEXT_FACTORY
url	PROVIDER_URL
username	SECURITY_PRINCIPAL
password	SECURITY_CREDENTIALS

The following are the steps to develop JNDI Application.

1. Create Hash table.
 2. Store the details in the Hash table using keys and values.
 3. Get the connection to directory server by supplying Hash table object as input.
 4. Call the methods bind/unbind/rebind/lookup methods to perform the operations with Directory server.
- Develop a JNDI application to store a string object into directory server.
To store an object in the directory server we use a method bind.

Syntax: void bind(key, Object)

Ex:

```
import java.util.*;
import javax.naming.*;
public class StoreObject{
    public static void main(String[] args) throws NamingException{
        Hashtable h = new Hashtable();
        h.put(Context.INITIAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFactory");
        h.put(Context.PROVIDER_URL,"t3://lacalhost:7001/");
        h.put(Context.SECURITY_PRINCIPAL,"kesav");
        h.put(Context.SECURITY_CREDENTIALS,"kesavchari");
        Context ic = new InitialContext(h);
        String name = "Raju";
        ic.bind("uname",name);
    }
}
```

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

- Demonstrate how to retrieve the data from directory server.

```
Context c = new InitialContext(h);
Object o = c.lookup("uname");
String s = (String)o;
```

```
System.out.println(s);
```

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

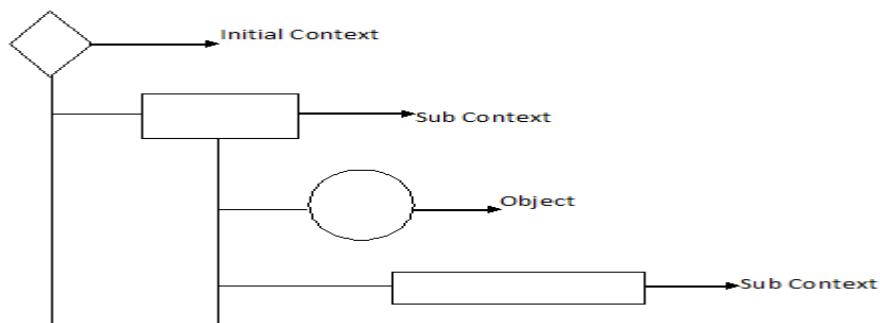
- We using a method rebind to update a Record a directory server.

```
Context c = new InitialContext(h);
c.rebind("uname","Naveen");
```

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

```
Context c = new InitialContext(h);
c.unbind("uname");
```

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



- To create the sub context use a method.

```
Context c = new InitialContext();
c.createSubcontext("Btech");
c.createSubcontext("mca");
```

- To create the sub context inside the sub context we can create the separate dot.

```
Context c = new InitialContext();
c.createSubcontext("Btech.1styear.mech");
c.createSubcontext("Btech.1styear.it");
c.createSubcontext("Btech.1styear.csc");
```

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

```
Context c = new InitialContext();
c.createSubcontext("btech");
```

```

c.createSubcontext("btech.1styear");
c.createSubcontext("Btech.1styear.mech");
c.createSubcontext("Btech.1styear.it");
c.createSubcontext("Btech.1styear.csc");

```

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

```

Context ic = new InitialContext();
ic.bind("betch.1styear.csc.kesav","abc");

```

- To remove sub context we use a method destroy sub context.

```

Context c = new InitialContext();
c.destroySubcontext("betch.1styear.csc");

```

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

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

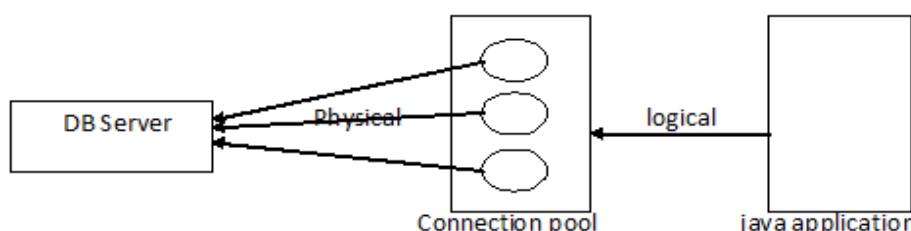


If we got a physical connection and we are not closing the connection after completion of work, the other application can not use the same connections.

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

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

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



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

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

```
import org.apache.commons.dbcp.*;
import java.io.*;
import java.sql.*;
public class DBC3P0{
    public static void main(String args[])throws IOException{
        BasicDataSource bds = new BasicDataSource();
        bds.setDriverClassName("oracle.jdbc.driver.OracleDriver");
        bds.setUrl("jdbc:oracle:thin:@localhost:1521:xe");
        bds.setUsername("system");
        bds.setPassword("admin");
        bds.setInitialSize(3);

        Connection con1 = bds.getConnection();
        System.out.println(con1);
        System.in.read();
        System.in.read();

        Connection con2 = bds.getConnection();
        System.out.println(con2);
        System.in.read();
        System.in.read();

        Connection con3 = bds.getConnection();
        System.out.println(con3);
        System.in.read();
        System.in.read();

        Connection con4 = bds.getConnection();
        System.out.println(con4);
        System.in.read();
        System.in.read();

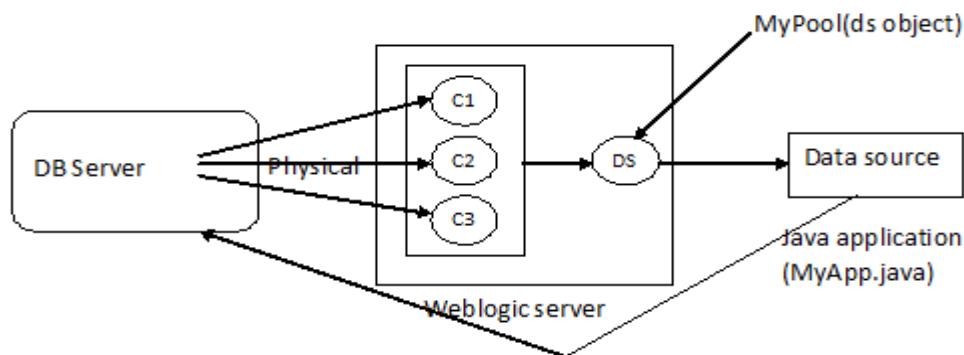
        Connection con5 = bds.getConnection();
        System.out.println(con5);
        System.in.read();
        System.in.read();
    }
}
```

```

import com.mchange.v2.c2p0.*;
import java.sql.*;
public class C3p0Connection{
public static void main(String[] args) throws Exception{
ComboPoolDataSource cpds = new ComboPoolDataSource();
cpds.setDriverClass("oracle.jdbc.driver.OracleDriver");
cpds.setJdbcUrl("jdbc:oracle:thin:@localhost:1521:xe");
cpds.setUser("system");
cpds.setPassword("malli");
cpds.setInitialPoolSize(3);
Connection con = cpds.getConnection();
System.out.println("Connection:" +con);
System.in.read();
System.in.read();
}
}

```

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



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

As part of the admin console go to → Domain Structure → services → JDB → Data Source.

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

Ex:

Name:	BMS Data Source
JNDI:	MyPool
Database Type:	oracle
Database Driver:	Oracle thin driver
NFXT	

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

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

Ex:

Database Name:	xe
Host Name:	local host
Port No:	1521
DB User Name:	system
Conform Password:	malli
NEXT	

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

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

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

```
import java.util.*;
import javax.naming.*;
import java.sql.*;
import javax.sql.*;
public class DBConnect2{
public static void main(String[] args) throws Exception{
Hashtable h = new Hashtable();
h.put(Context.INITIAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFactory");
h.put(Context.PROVIDER_URL,"t3://lacialhost:7001/");
h.put(Context.SECURITY_PRINCIPAL,"admin");
h.put(Context.SECURITY_CREDENTIALS,"inetsolv");
Context c = new InitialContext(h);
Object o = c.lookup("mypool");
DataSource ds = (DataSource)o;
Connection con = ds.getConnection();
```

```

System.out.println(con);
System.in.read();
System.in.read();
}
}

// set CLASSPATH=wlclient.jar;;
// set CLASSPATH=c:\bea\wlserver_10.3\server\lib\weblogic.jar;;
// echo %CLASSPATH%
// echo %PATH%
//c:\bea\user_projects\domains\mydomain\bin\setDomainEnv.cmd
// c:\bea\user_projects\domains\mydomain\cd\
To set the class path to "weblogic.jar;;" we can use "set DomainEnv.cmd".
```

*Retrieve the Records program?

```

import java.util.*;
import javax.naming.*;
import java.sql.*;
import javax.sql.*;
public class RetrieveRecords1{
public static void main(String[] args) throws Exception{
Hashtable h = new Hashtable();
h.put(Context.INITIAL_CONTEXT_FACTORY,"weblogic.jndi.WLInitialContextFactory");
h.put(Context.PROVIDER_URL,"t3://localhost:7001/");
h.put(Context.SECURITY_PRINCIPAL,"admin");
h.put(Context.SECURITY_CREDENTIALS,"inetsolv");
Context c = new InitialContext(h);
Object o = c.lookup("mypool");
DataSource ds = (DataSource)o;
Connection con = ds.getConnection();
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from emp");
while(rs.next()){
System.out.println(rs.getString(1));
System.out.println(rs.getString(2));
System.out.println(rs.getString(3));
}
}
```

}

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

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

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

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

*****SERVLETS*****

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

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

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

All the applications which we have developed as of know standalone application.

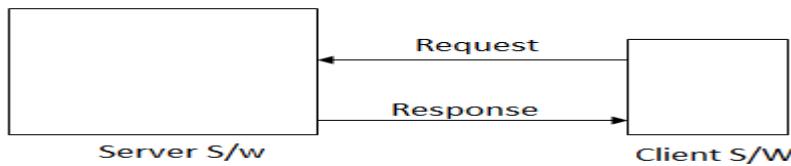
Standalone application will have a “main()” method.

Standalone applications run on their own. They are dependent on other program. Where as web based application are dependent on “Servers”.

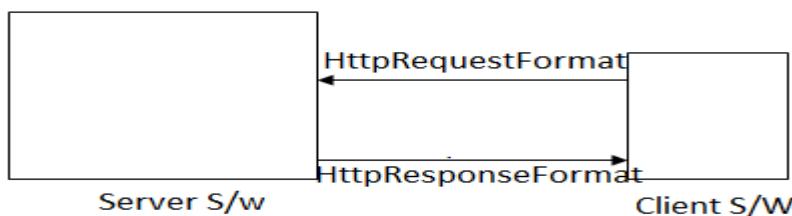
***Disadvantages:**

- A. As the standalone application Run’s on client computer, we need to install it in all the computers.
- B. We will have lot of maintainence related problems as part of the project. That is when ever to change the project we have placed with in all the client computers.
- C. A Standalone application consumes the resources of client computer (Hard disk, Processor and etc).
- D. Standalone applications are person independent only some fellows know how to install and settings the required configuration.
- E. To develop the web based applications meagerly two programs are required.
 1. Server
 - 2.Client

- F. The client sent to request the server takes the request and processes it sent the response to client.



- G. They are so many server software's are available some of them are weblogic, tomcat, websphere, jboss, resign , glash fish etc.
- H. They are so many client software's are available some of them are IE, chrome, firefox, opera, safari and netscape etc.
- I. As a developer we are responsible to develop a program which Run's inside the Server.
- J. To develop the web based applications we required "http" protocol. (Server and client those are same Protocol using with http)
- K. http protocol is devided into two parts. They are:
1. http Request format
 2. http Response format



The following are http Request format and Response format.

<u>http Request format</u>	<u>http Response format</u>
Initial Request format	Initial Response format
0 or more no.of headers	0 or more no.of headers
Blank line	Blank line
Request body(optional)	Response body(optional)

***Initial Request Line:**

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

Method	Requested Resource	Protocol/Version
--------	--------------------	------------------

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

The following are method of http protocol. They are:

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

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

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

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

GET	/fout.html	HTTP/1.1
-----	------------	----------

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

The following is header format.

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

The following are the more important of two headers.

1. User-Agent
2. Accept-Language

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

To represent a next line character we use CR LF.

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

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

***Initial Response Line:**

Initial Response line is divided into three parts.

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

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

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

HTTP protocol guys are given couple of status code.

100 to 199 Information

200 to 299 Successes

300 to 399 Re-Direct

400 to 499 Request Resources not available

500 to 599 Failed to execute Request Resource

***Status Message:** every status code will have status message.

Ex: 200 – Ok

404 – Not Found

When the server send 2xx status code we can understand that server is able to process.

The request sends by the client properly.

When we get 404 or 4xx we can understand that the Request Resources is not available in server. When we get 5xx the Resource is available with server but failed to process the resource.

3xx is indicates re-direction to other servers. An HTTP Response header indicates server sending extra information to client (server →client).

The most important header send by the server is “Content type: text/html”. Based on the context type client will render output to the user. It is the responsibility of the client to rend/display output to the client.

- Method indicates which operation has to be carried out by the server.

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

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

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

This URL is called as Query String

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

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

Ex: post /two.html HTTP/1.1

Accept-language: en-us

uname=raju&pwd=mypassword

*What is different between GET and POST methods?

In case of get() method request the data will be appended to url and send to server.

Where as from the post request the data will be appended to request body.

It's not recommended to use get() to transfer sensitive data.

This is a limitation of 1024Kb size to transfer the data by using get() method. This is no limitation of transferring the data by using post() method.

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

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

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

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

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

Ex: 1xx information

3xx redirect

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

1. State less protocol
2. State full protocol

***State less protocol**: A state less protocol can't remember all the conversations which is happening with client.it will remember Lasr conversation.

Ex: HTTP Protocol

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

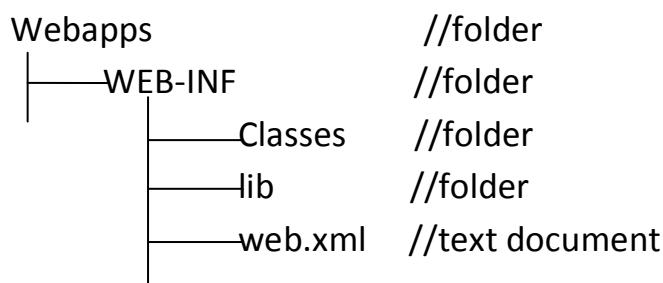
Ex: smtp, sntp, tcp/ip and etc.

*****Procedure to develop weblogic application:**

1. Create a folder who's folder name must be project name (there should not spaces).
- Note:** we call this folder as project context/web context.
2. Inside the above created folder we create a folder with the name "**WEB-INF**" (must be write the only in capital letters).
3. Inside the "**WEB-INF**" cerate classes and lib folders.
4. Create a file whose file name must be "**web.xml**" (inside WEB-INF).



***Project Structure:**



1. By using java we can develop both static and dynamic web-based applications.

2. We can develop two types of web-based applications. They are:

- 2.1. Static web-based applications
- 2.2. Dynamic web-based applications

3. What are static web-based applications?

When the client send the request to the server, if the server is sending same output for multiple times. Then we call as static web-based applications. To develop static web-based application we use HTML, java script and images.

4. What are dynamic web-based applications?

If the output is getting change the every time we call the applications as dynamic web-based applications we use servlets, JSP's to develop dynamic web-based applications.

5. *“WEB-INF” folder is called as a “private” folder this is because the file in this folder can be accessible only by the server.

6. web.xml file is called as “Deployment descriptor”.

7. *All the html files of our project must be placed inside “project and outside WEB-INF” folder.

8. Once if we are ready with the project, we must deploy the project.

9. **what is the meaning of Deployment?

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

10. By default tomcat server uses 8080 port number in tomcat we use webapps folder as the Deployment folder.

11. To deploy a project it's the recommended to starts the server and place the project in webapp (deployment) folder.

<http://localhost:8080/inetsolv/one.html> or web.html

12. To deploy a project in weblogic server we use a folder auto deploy (weblogic server).

13. Most of the times we need to read deploy the project in the server.

*What is the re-deployment?

Removing the existing project and deployment the project is called as re-deployment.

In case of java web-based applications the web applications are called as portable. We can deploy the project on any server.

In windows operating system because security reasons they are provided a UAC (User Access Controls) controls

. To disable it start control panels user accounts
—————> Change user account control settings.

Note: Re-Start

In tomcat when we see an error message address already in use or JVM_Bind the problem is because of some other program is already using the same port number by change the port number of tomcat server we can re-solve it.

We can change the port number of tomcat server manually by going to "server.xml" file. We can find this xml file as part of confi folder.

Tomcat server is released in the form of .zip file also. To run the tomcat S/W from the zip file are use "startup.batch".

Tomcat S/W is dependent an on Enviroment variable.

*Servlet API is used to develop webbased application.

The following are the most Important "**packages**" to develop servlets. They are:

1. Javax.servlet
2. Javax.servlet.http

The following are the most important interfaces and classes of javax.servlet packages.

Javax.servlet:

Interface:

1. Servlet
2. ServletRequest
3. ServletResponse
4. ServletConfig
5. ServletContext

Classes: genericServlet

The following are the most important interfaces and classes of javax.servlet.http packages.

Javax.servlet.http:

Interfaces:

1. HttpServletRequest
2. HttpServletResponse

Classes: HttpServlet

***Servlet:**

The servlet is an API , Released by SunMicro System. The servlet is a java program which runs inside the Server.

The servlet is a java program which provides the 'implementation' of servlet interface directly or indirectly. The following is UML diagram of servlet interface.

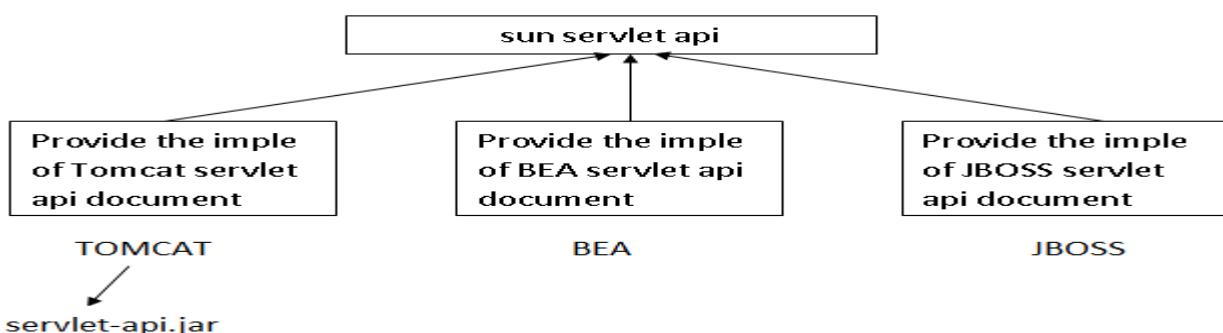
javax.servlet.Servlet (package and interface)
void init(ServletConfig)
void service(ServletResult,ServletResponse)
void destroy()
ServletConfig getServletConfig ()
String getServletInfo()

***Procedure to develop first servlet program:**

Step 1: Develop a class which provides the implementation of Servlet Interface.

```
public class FirstServlet implements Servlet{
    ServletConfig config;
    public void init(ServletConfig config){
        this.config = config;
        System.out.println("we are in FirstServlet init() method");
    }
    public void service(ServletRequest request, ServletResponse response){
        System.out.println("we are in FirstServlet service() method");
    }
    public void destroy(){
        System.out.println("we are in destroy() method");
    }
    public ServletConfig getServletConfig(){
        return config;
    }
    public String getServletInfo(){
        return "this is my First servlet";
    }
}
```

Step 2: compile the FirstServlet program. To compile the servlet program we have set the class path to a jar file which provides the implementation of servlet api. The name of the jar files vary between company to company in case of tomcat server the name of the jar file which provides to implementation of "servlet-api.jar".



Step 3: create a web-based application.

Step 4: configure the servlet into above web-based application.

Step 4.1: copy all servlet related .class files into classes folder.

Step 4.2: configure servlet into web.xml file.

```
<web-app>
<servlet>
    <servlet-name>a</servlet-name>
    <servlet-class>FirstServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>a</servlet-name>
    <url-pattern>/fs</url-pattern>
</servlet-mapping>
```

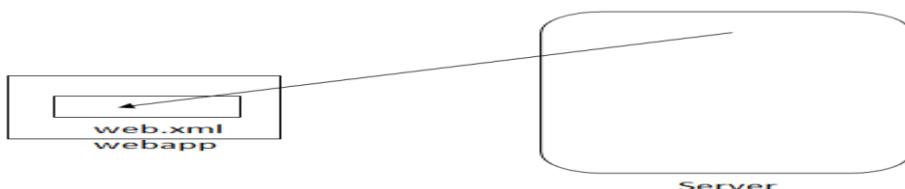
Step 5: Deploy the project in any server.

Step 6: The following url which is used to access the first servlet.

<http://localhost:8000/webapp/fs>

Step 7: It's mandatory that every servlet must be configure in "web.xml" file. When we deploy the project server reads the contents from 'web.xml' file.

Step 8: when we deploy any web-based application server will search for web.xml file. If the server found the xml file it will read the contents from web.xml file and store it in JVM's memory.



When we observe the above behavior it tomcat server. If it doesn't found web.xml it's not displaying any error message. Where as the web logic server has displayed an error message invalid application type.

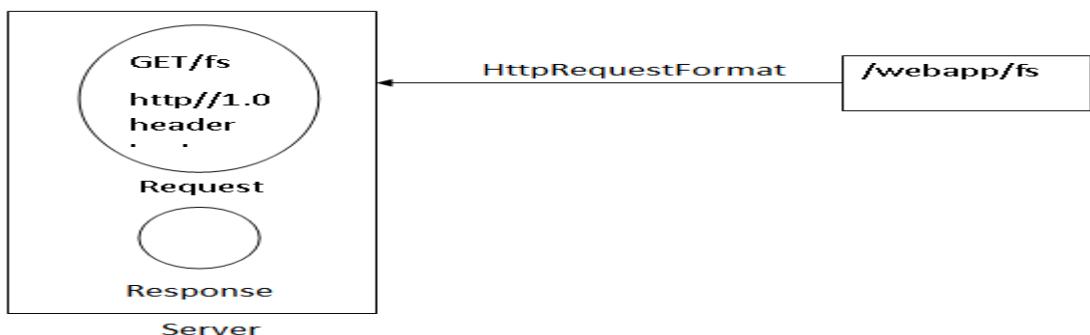
To read the contents from xml files are required "parser" programs. In java they are two parser are available.

1. SAX parser
2. DOM parser

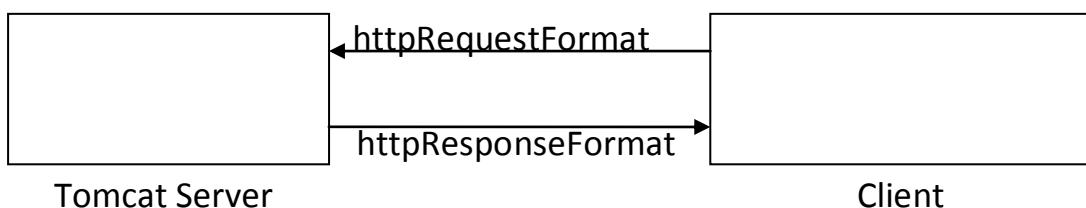
When deploy a project if the xml file is valid xml file server reads the contents and store it in JVM's memory. If the xml file is invalid SAX parser program throws an error message. **When we develop a project, server will not create any servlet objects.**

Tomcat Server

When the client sends the request to server, server creates two objects. They are request object and response object. Now the server opens http request format and store the data into request object.



It is the responsibility of server to remove request and response objects after finishing the processing of resource.



After the client has sent the request to server, server opens request and get the requested resource. Now the server checks is there any url pattern configured in web.xml file for that requested resource.

If it is available if does the remaining work. If it is not available in web.xml file server sends httpResponseFormat by using a status code you.

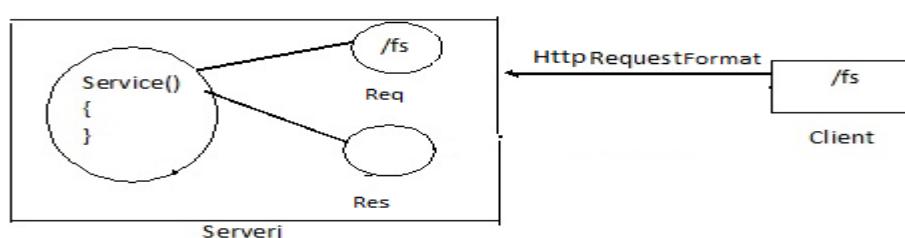
If the resource is available in web.xml file, server gets the servlet class name. Now the server checks is there any servlet object is available in the server. If not available it is the responsibility of server to create servlet object.

The server will try to load the class from class's folder. If the class file is not available in the classes folder, server throws an Exception java.lang.classNotFoundException.

If the class is loaded successfully server creates first servlet object and call the init().

Note: init() will be called only once when the servlet object is created.

Now the server will status the Exception service() to Execute the service() server supply request and response objects as parameters.



After the server has executed service() if it is the responsibility of server to send the response back to client. When the server sends the response to client, server removes request and response objects. The FirstServlet object remains same in the server. It is the responsibility of server to remove servlet object. In the following two scenarios server will remove servlet objects.

1. When we stop the server.
2. When we undeploy the project.

When we got an Exception illegal access specifiers the problem is because the servlet class is not a public class or the constructor is not a public. If we got all Exception classxxxservlet is not a servlet. The problem is because of the class doesn't provide the implementation of servlet interface.

- **JDK is used by all the developers to develop the program and test it(JDK contains javac, java, javadoc, jar and etc).**
- **JRE is use to run any java based applications. Most of the time customers install only JRE to run java based applications.**

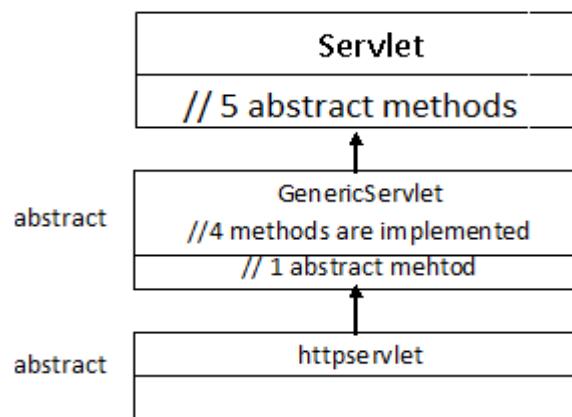
When we got an Exception saying “unsupported class version” the problem is because of our java compiler version and JRE version used by the servers.

To resolve unsupported class version, we used to make that java compiler version and JRE version of server are same.

Init(), service(), destroy() are called as servlet life cycle methods. In servlets the business logic will be written as part of service(), any code which has to be Executed only one time when the server create servlet object is provide in the init() method.

Any code which has to be executed only once at the time of servlet object is removed. Then we provide the code in destroy() method.

To simplify the development of servlets sun micro systems has given predefined classes. They are generic servlet, http servlet. The following diagram shows the relationship or inheritance between these classes.



Different people says there are 3 ways are there to develop the servlets. According to sun micro system it is always recommended to use http servlet class to develop the servlet. **If we develop the servlets based on HttpServlet we can remove the redundant code of init(), destroy(), getServletConfig(), getServletInfo() and etc.**

The following is the servlet based on HttpServlet:

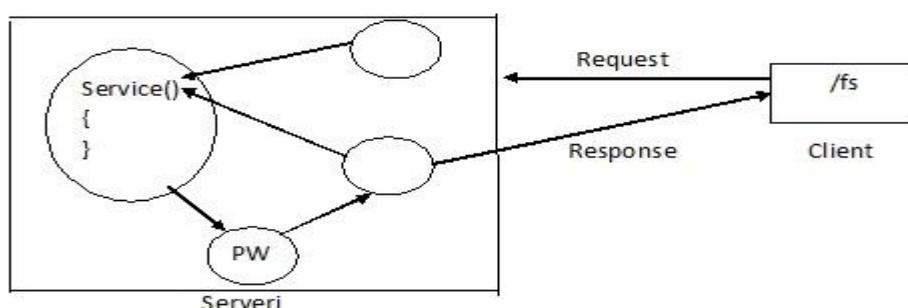
```
Public class FirstServlet Extends HttpServlet {  
    Public void service(HttpServletRequest request, HttpServletResponse response){  
        System.out.println("we are in service() method of FirstServlet");  
    }  
}
```

The servlets can be used to develop two types of applications. They are:

1. Static Servlet
2. Dynamic Servlet

As part of the servlets when we use System.out.println() it is Executed on the server and display the message on the server. But for the web-based applications we have to send the output to client.

It is the responsibility of server to take the data from response object and convert it into httpResponseFormat and sent to client. **To send the output to client we have to add the data to response object. To add the contents to response object we take the help of print writer object.** The following diagram shows how output will be sent to client.



```
Public class FirstServlet Extends HttpServlet{  
    public void service(HttpServletRequest request, HttpServletResponse response){  
        PrintWriter p = response.getWriter();  
        p.print("welcome to");  
        p.write("servlets");  
    }  
}
```

The following is the servlet which send html contents to the client. To send the html context to the client we have to write html tags as part of write() method or print() method.

```
public class WelcomeServlet Extends HttpServlet{  
    public void service( ----- ){  
        PrintWrite out = response.getWriter();  
        out.Println("<html>");  
        out.Println("<tittle>MyProc</title>");  
        out.Println("<body>Naidu SMA</body>");  
        out.Println("</head></html>");  
    }  
}
```

The above servlet is trying to send html content to the client. Every time when the client send the request it is displaying same output. These type of applications are called as static web based applications.

By using servlet also we can develop static applications as well as Dynamic applications. By using servlets it is not recommended to develop static web based applications. It's always recommended to develop Dynamic web based applications.

Instead of writing printwriter object we can use "Servlet.OutputStream".

It is recommended to use ServletOutputStream to send binary data.

Ex: ServletOutputStream out = response.getOutputStream();
 out.Println("Kesav");

The following are disadvantages of developing static servlets.

1. It takes huge amount of time for developing the project.
2. If you want to change the project we need to identify the code and do the necessary changes. Because of this reason we get maintenance related problems.
3. It occupies lot of resources of server. Because of all these reasons it's not recommended to develop static servlets.

Instead of developing a servlet which send the static content recommended to develop html files to send the static content. The following is an example of Dynamic web-based application which display corrent date and time.

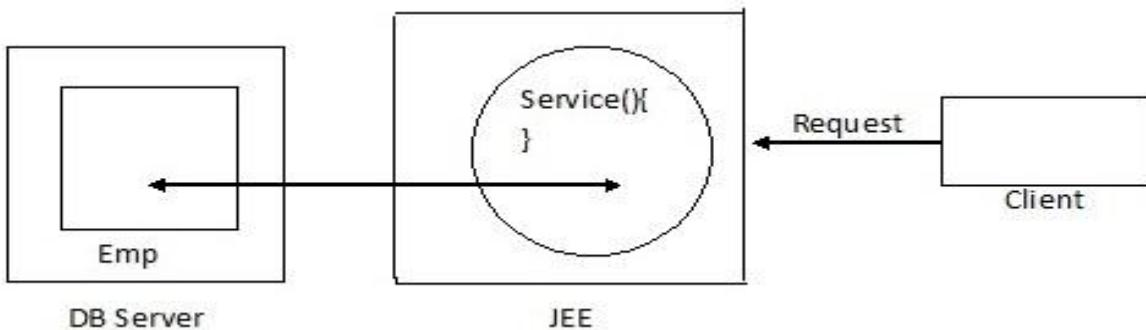
```
public class DateAndTimeServlet Extends HttpServlet{  
    public void Service( ----- ){  
        Date d = new Date();  
        PrintWriter out = response.getWriter();
```

```

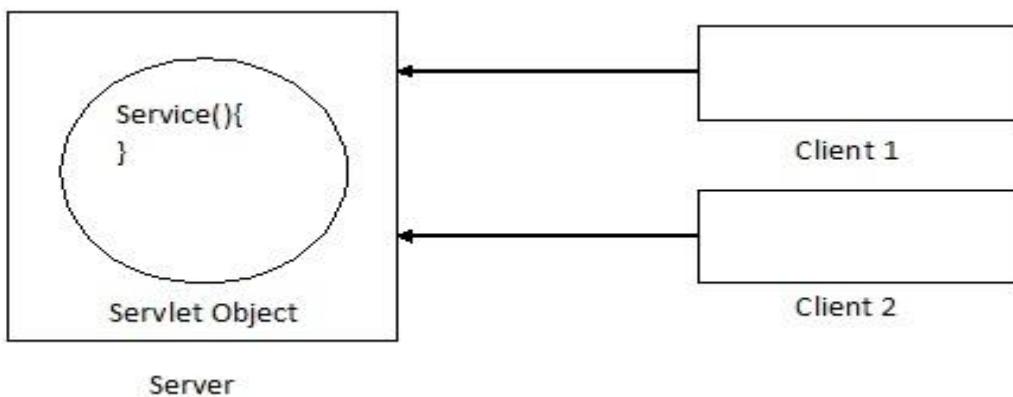
        out.Println(d);
    }
}
}
}

```

Develop a servlet which retrieve the data from data base server and send the output to the client.



When multiple clients try to access one servlet only one servlet object will be created. This servlet object is shared between multiple clients.



When we got an error message like “HTTP status 405-HTTP method GET is not supported” we can understand that service() method is not available.

The following is the servlet code which retrieve the data from data base server and display to the client.

```

public class RetrieveRecordsServlet Extends HttpServlet{
    public void service( ----- ){
        try{
            // standard JDBC code
            ResultSet rs = stmt.executeQuery("select * from product");
            PrintWriter out = response.getWriter();
        }
    }
}

```

When we deploy a project and if the server want to create the objects or server want to use the classes it will check in the following order.

1. Server checks for class files in classes folder.
2. If it is not available in classes folder server checks for project lib folder.
3. If it is not available in project lib folder, server check in server lib folder. If it is not available in these entire places server throws an Exception NoClassDefFound.

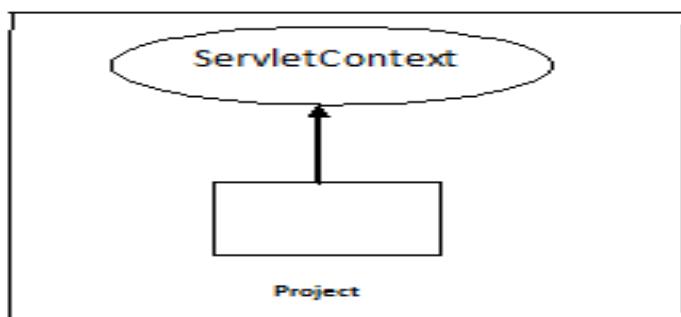
Develop a servlet and create a jar file and deploy the project and test it. In the above Retrieve Data Servlet we are hard coded driver class, url, username and password. Because of this reason the servlet able to communicate with only one Data base server. We would like to remove the hard coding from above servlet.

To secure the hard coding we are try to use **command line arguments and system properties**. But these will **not work in servlets**. This is because we can't supply command line arguments or system properties to the server.

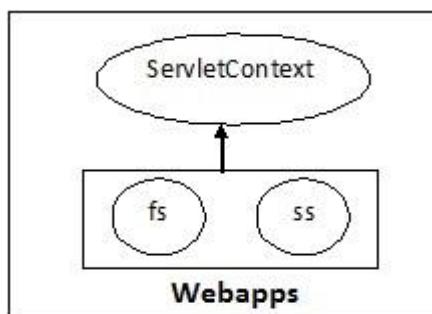
As the server is running the servlet application we can't use command line arguments or system properties.

ServletContext, ServletConfig objects are used to remove hard coding. These objects are managed by servers(creation and removing the objects).

ServletContext: It is the responsibility of server to create ServletContext object. It is the responsibility of server to remove ServletContext object. **Server creates the ServletContext object at the time of project is deployed. Per project we will have only one ServletContext object.**

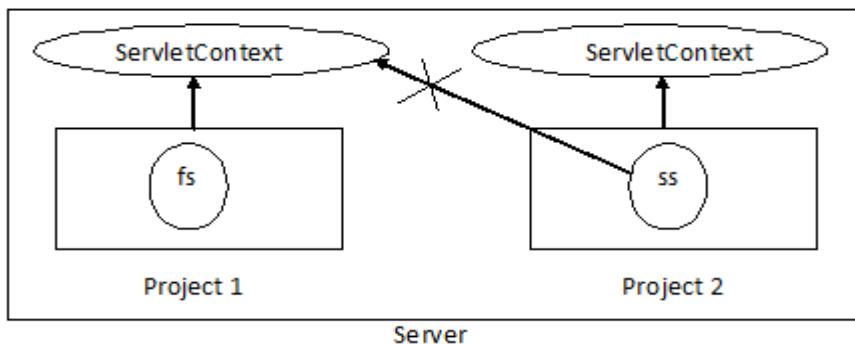


In a server we will have multiple ServletContext objects. This is based on number of projects available in the server. We can store the data into ServletContext object. The data stored in the ServletContext object can be accessible by all the resources of that project.

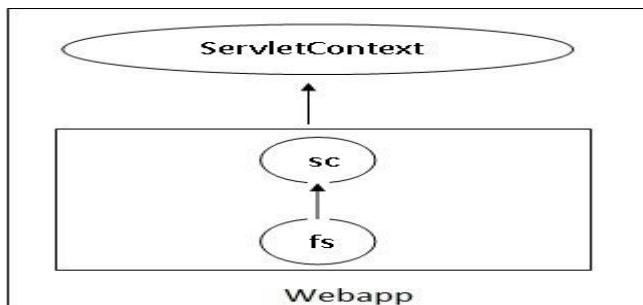


From the above diagram if we store the data into ServerContext object. FS and SS servlets can access the data.

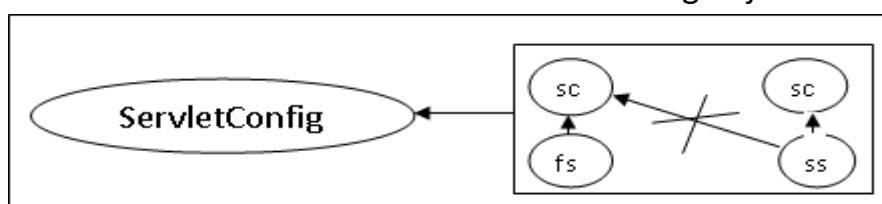
If we store the data into one ServletContext object the other project can't access the data from current project.



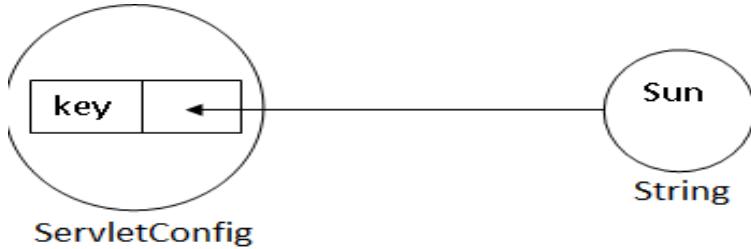
ServletConfig: It is the responsibility of server to create ServletConfig object. It is the responsibility of server to remove ServletConfig object. **Server is creating the ServletConfig object when ever the servlet object is created. Server removes the ServletConfig object when ever servlet object is removed.**



In a server we will have N no.of ServletConfig objects. It's all based on the no.of servlets. If we store the data into ServletConfig object only that servlet can access it other servlets can't access the data from this ServletConfig object.



As a developer we can't write the code to create ServletContext and ServletConfig objects. To use these objects we have to provide the code to get the reference of these objects. To store the data into all object we required key values. This is useful to store the data as well as to retrieve the data from any object.



GenericServlet class provides the implementation of two interfaces servlet and ServletConfig. In this class there is a method getServletConfig() which return ServletConfig object.

```

class GenericServlet implements Servlet, ServletConfig{
    ServletConfig getServletConfig(){
        return config;
    }
    -----
}

```

Procedure to use ServletConfig object:

1. Store the data into ServletConfig object by using init parameter as shown below.

```

<web-app>
    <servlet>
        <servlet-name>ts</servlet-name>
        <servlet-class>TestServlet</servlet-class>
        <init-param>
            <param-name>uname</param-name>
            <param-value>bms</param-value>
        </init-param>
    </servlet>
    -----
</web-app>

```

2. To read the data from ServletConfig we use a method getInitParameter().

```

public class TestServlet implements HttpServlet{
    public void Service(----- ){
        ServletConfig config = getServletConfig();
        System.out.println(config.getClass());
        String uname = config.getInitParameter("uname");
        String pwd = config.getInitParameter("pwd");
        PrintWriter out = response.getWriter();
    }
}

```

```
        out.println(uname);
        out.println(pwd);
    }
}
```

If the data is not available in the ServletContext object with the given key it returns null value, to read the value from ServletConfig object we use the method getInitParameter().

Syntax: String getInitParameter(key);

If the specified key is available in ServletConfig object we get the value in the form of string. If the key is not available we get null value. To specify multiple InitParameters we must use multiple InitParameters tags as shown below.

```
<servlet>
    <servlet-name>ts</servlet-name>
    <servlet-class>TestServlet</servlet-class>
<init-param>
    <param-name>uname</param-name>
    <param-value>bms</param-value>
</init-param>
<init-param>
    <param-value>pwd</param-value>
    <param-value>abc</param-value>
</init-param>
</servlet>
```

We have developed another servlet to read the data from TestServlet config object. When we executes this program we got the null value. If we don't know the key values we can get the key names available in ServletConfig object by using getInitParameterNames().

The following is an example of get the key names and display to the client.

```
public class TestServlet Extends HttpServlet{
    public void Service( ----- ){
        PrintWriter out = response.getWriter()
        ServletConfig config = getServletConfig();
        Enumeration e = config.getInitParameterNames();
        while(e.hasMoreElements()){
            out.println(e.nextElement() + "\n");
        }
    }
}
```

```
    }
}
```

Procedure to use ServletContext object.

1. To store the data into ServletContext object we use the tag <context-param> in web.xml file as shown below.

```
<web-app>
  <context-param>
    <param-name>uname</param-name>
    <param-value>bms</param-value>
  </context-param>
  <context-param>
    <param-name>pwd</param-name>
    <param-value>abc</param-value>
  </context-param>
<servlet></servlet>
-----
-----
</web-app>
```

2. Get the ServletContext objects. There are 2 ways are available to get the ServletContext object.
 - 2.1. By taking the help of ServletConfig object.
 - 2.2. We can call a method directly from GenericServlet class.

Ex: `ServletConfig config = getServletConfig();`
`ServletContext application = config.getServletContext();`

OR

```
ServletContext application = getServletContext();
```

3. To read the values from context object we use a method `getInitParameter()`.

```
public class TestServlet Extends HttpServlet{
public void Service( ----- ){
PrintWriter out = response.getWriter();
ServletConfig config = getServletConfig();
ServletContext application = config.getServletContext();
String uname = application.getInitParameter("uname");
String pwd = application.getInitParameter("pwd");
String url = application.getInitParameter("url");
out.println(uname);
out.println(pwd);
```

```
    out.println(url);
}
}
```

We have developed another servlet to read the data from ServletContext object. In this servlet instead of service() method we are use a method doGet() method.

```
public class AnotherServlet Extends HttpServlet{
    public void doGet(HttpServletRequest request, HttpServletResponse response){
        // code to get the data from the ServletContext and display to the client
    }
}
```

When we configure the above servlet we are use the url pattern as *.a.

```
<servlet-mapping>
    <Servlet-name>as</servlet-name>
    <url-pattern>*.a</url-pattern>
<servlet-mapping>
```

Develop a servlet to retrieve the data from DB server and display the output to the client. To achieve this we required to use weblogic connection pool and test it in both tomcat and weblogic servers. We would like to develop a servlet which send the following output.

First Line
Secod Line

To achieve the above requirement we have used println() as part of service() method.

```
public void doGet( ---- ){
    PrintWriter out = response.getWriter();
    out.Println("First Line");
    out.println("Second Line");
}
```

When we run the above application we got different outputs in different browsers. As the browser understand any HTML tag as part of the response we have added
.

```
    out.Println("First Line <br>");
```

When the above line is Executed IE is able to process the
 where as other browsers are displaying
 directly to the client. To resovle the above problem we have to specify context type.

Syntax: void setContextType(String type);

Type attribute specify what type of context is sent by server to client. The following are some of the ContextTypes.

text/html , text/xml, application/pdf, application/excel, ----- etc.

ContentType are also called as MIME types to send the xml file to the client we have specify the ContextType as text/xml. To send the error to the client we use a method response.SendError() method.

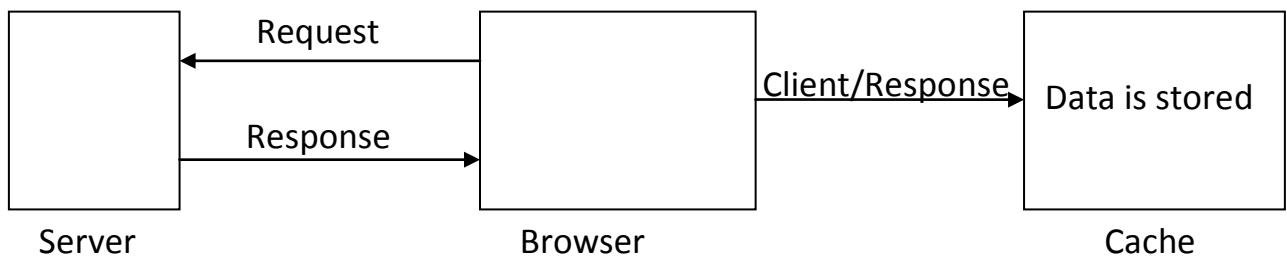
Ex: response.SendError(555,"some problem");

When an Exception is occurred, the servlet control goes to catch block. If we doesn't write any code in the catch block use all see a blank server. It is not recommended to display the blank server to the user. Always we have to display an error message to the client. To achieve this we use a method response.SendError().

Ex: response.SendError(543,e.getMessage());

It is always recommended to use SendError() in catch block. Because of competition b/w the browser and every browser will release lot of features. The server like IE and Firefox use browser cache mechanism.

Now days all the modern browsers are associated with browser cache.



When the server is sending output to the client by default the expire data to the context is set as none. If express is none browser will never send the second request to the server. This is because before sending the request to server browser check whether the data is available in the cache or not. If it is available and if it is not Expired browser will not send therequest to server.

Ex: public class RRServlet Extends HttpServlet{
 public void Service(----){
 response.setHeader("Expires",System.currentTimeMillis()+7200000);

 }
}

To resolve all the browsers cache related problems we use a header "cache-control". When by using this header we can make sure that browser doesn't catch the data. The

advantage of this approach is every time. When the client send the request, that request go to server.

Ex: public class
 public void
 response.setHeader("Cache-Control","no-cache");

 }
}

When ever we develop any web-based application the first two lines of web-based application must be content type and setHeader.

Form Based Applications: Most of the web developer's responsibility is to develop form based applications. **FormBased App's are responsible to capture the data from the user and send the data to server.** To work with form based applications we should know how we call the resources available in server.

Senario-1: We can call the servlet **by typing the url in the browser.**

Senario-2: From the form we can call servlets **by using submit button.**

Ex: <form action = "/webapp/ws">
 <input type = "submit" value = "callsevlet"/> </form>

When the user click on submit button browser get action attribute value and converted it into HttpRequestFormat and send to server.

Senario-3: **The Anchor tag** following is a tag which is use to call a servelt from Anchor tag. call servlet

Using user defined buttons: In HTML we can use **user defined buttons**. When we use these buttons we have to provide the javascript to submit the form. The following is an example.

```
<script>
    Function doWork(){
        window.document.form[0].action = "/webapp/ws";
        window.document.form[0].submit();    }
</script>
<form onclick = "doWork()">
    <input type = "button" value = "call servlet"/>
</form>
```

Prototype: **Prototype is a like a dummy project.** When we deliver a prototype to a customer by looking at the prototype customer understand now the final project looks

like. Before the project coding starts we develop the prototype and deliver to customer. The following is the prototype of the form based application which we want to develop.

The diagram shows a rectangular form with three components: a 'UserName' label next to a text input field, a 'password' label next to a text input field, and a 'login' button below them.

To implement the above requirement we have to develop two programs. They are:

1. Program to display the form.
2. Program to get the data from the client and perform the operations.

The following are the two approaches to implement the above requirement.

Ap-1:

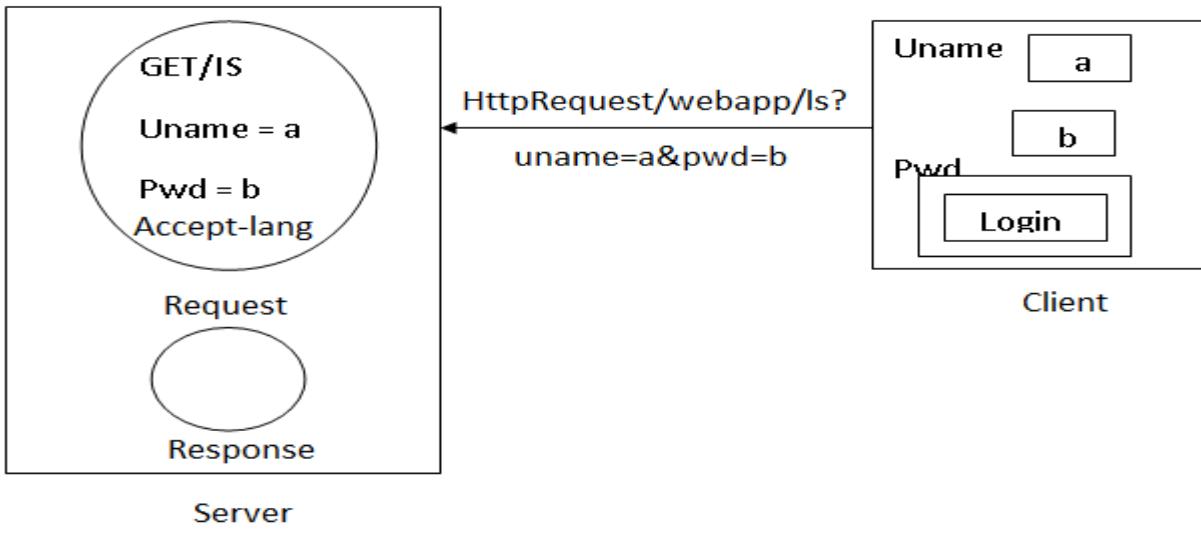
- | | |
|-----------------------|---|
| 1. DisplayFormServlet | - DisplayForm |
| 2. ProcessFormServlet | - get the data from the client and communicate with DB. |

Ap-2:

- | | |
|-----------------------|--|
| 1. login.html | -DisplayForm |
| 2. ProcessFormServlet | -get the data from the client and communicate with DB. |

When the user files the form and click on submit button the following steps will be carried out.

1. Browser captures the data and stores the data into the variables which are allocated to the fields.
2. Browser gets action attribute value. Based on the data the url as shown below.
/webapp/ls?uname=a&pwd=b
3. Now the browser creates httprequestformat and sends the request to server.
4. Now the server creates request and response objects. It is the responsibility of server to read all the data and place it in request object. When we supply this request object as input to servlet it can read all the details.



To implement the above requirement we are implemented the following two programs.

//Login.html

```

<html>
-----
<form action = “/webapp/lst”>
    username:<input type = “text” name = “uname”/><br>
    password:<input type = “password” name = “pwd”/><br>
    <input type = “submit” value = “Login”/><br>
</form>
</html>

```

The following is a servlet which captures the data form.

```

public class LoginServlet Extends HttpServlet{
public void Service( ----- ){
String uname = request.getParameter(“uname”);
String pwd = request.getParameter(“pwd”);
String address = request.getParameter(“address”);
response.setContentType(“text/html”);
PrintWriter out = response.getWriter();
out.Println(uname+“);
out.Println(pwd+“);
out.Println(address+“);
}
}

```

To transfer the data between client and server we can use either a `get()` method or `post()` method. In case of `get()` the data will be appended to url. In case of `post()` the

data will be appended to request body. Most of the projects uses the post() to transfer secured data.

The following is a syntax of request.getParameter

```
String getParameter(fieldname);
```

request.getParameter() returns a null value if the specify field name is not available. A form based application to get the data from a drop down. //w3schools.com

As part of the servlets the methods like doGet() and doPost() methods. **doGet() will be Executed when ever we send a get request. doPost() will be Executed when ever send the post request. To support all the methods we use service()**. We have implemented the form based application based on application. In this we are used a servlet to send the form output to the client. The following code demonstrate how do we read the values of the form without aware of field names.

```
public class LoginServlet Extends HttpServlet{  
    public void Service( ----- ){  
        PrintWriter out = response.getWriter();  
        Enumeration e = request.getParameterNames();  
        while(e.hasMoreElements()){  
            String key =(String) e.nextElement();  
            out.Println(key);  
            out.Println(request.getParameter(key));  
        }  
    }  
}
```

The following code is used to get the map object which contains both keys and values.

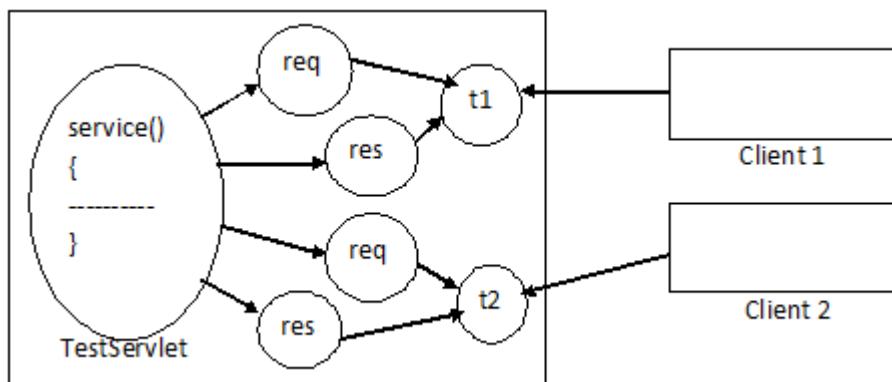
```
public class LoginServlet Extends HttpServlet{  
    public void Service( ----- ){  
        PrintWriter out = response.getWriter();  
        Map map = request.getParameterMap();  
        Set keys = map.keySet();  
        Iterator i = key.iterator();  
        while(e.hasNext()){  
            object o = i.next();  
            String s = (String)o;  
            out.println(s);  
        }  
    }  
}
```

A web-based application will be accessed by any no.of clients at the same time. If we develop a servlet at the same time two different clients can send the request to the server.

Every JEE server contains a thread pool. This thread pool program will be started when ever the server is started. When even a client sends the request to server picses a thread from thread pool and creates request and response objects. These objects are handover to service().

After the server has finished its work server will removes request object and response object and it return thread to thread pool.

The following diagram shows how two different clients are working on one servlet objects.



We have observed that a different server follows different algorithms to pick a thread from thread pool. **Can we have a constructor in a servlet? Yes** we can have a constructor. **We can't have parameterized constructors.** Why we can't have parameterized constructors in servlet **Every server uses class.forName() to create servlet object.**

When we have a parameterized constructor, class.forName() will fail to create an object by throwing an Exception java.lang.InstantiationException.

The following is an example code used by ever server to create the object.

```
public class TomcatServer{  
    public static void main(String args[]){  
        Class c = class.forName(args[0]);  
        Object o = c.newInstance();  
        System.out.println(o.getClass());  
    }  
}
```

We can't create a parameterized constructor as part of servlets. We can provide only default constructor. Because of this reason sun micro system is not recommending to

use constructors in the servlet. As an alternative to the constructor sun micro systems has provided init().

As part of the init() we have to provide the code which has to be Executed only once when the object is created.

The following is the code which we found in every server. How they creating the objects and calling the methods.

```
public class TomcatServer{  
    public static void main(String args[]){  
        Class c = Class.forName(args[0]); //class Name is read from web.xml file  
        Object o = c.newInstance();  
        Servlet s = (Servlet)o;  
        s.init();  
        s.service();  
        s.destroy();  
    }  
}
```

When the server create servlet objects always server call the init(). First server check whether init() available in our class or not. If not available server goes to super class and check for the methods.

By using our own techniques we can make sure that server call parameterized constructor from default constructor.

Ex:

```
public class TestServlet implements Servlet{  
    public TestServlet(){  
        this(10);  
        System.out.println("TestServlet object is created default");  
    }  
    public TestServlet(int dummy){  
        System.out.println("TestServlet object is created");  
    }  
}
```

When the server is creating servlet object:

Scenario-1: When the client sends the request to the server for the first time server is creating servlet object.

Scenario-2: We can use load-on-startup. We are configure<load-on-startup> in web.xml file as show below.

```

<web-app>
    <servlet>
        <servlet-name>a</servlet-name>
        <servlet-class>FirstServlet</servlet-class>
        <load-on-startup>5</load-on-startup>
    </servlet>
    -----
</web-app>

```

The `<load-on-startup>` takes a +ve value as input. This +ve value indicates the priority to create the servlet object.

Ex:

```

<web-app>
    <servlet>
        <servlet-name>a</servlet-name>
        <servlet-class>FirstServlet</servlet-class>
        <load-on-startup>5</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>b</servlet-name>
        <servlet-class>SecondServlet</servlet-class>
        <load-on-startup>2</load-on-startup>
    </servlet>
    -----
</web-app>

```

We have observed that for two different servlets we have given the same priority. When we deploy the project in TomcatServer he has created the object to both FirstServlet and SecondServlet. But TomcatServer is using an algorithm based on the name of the servlet (based on alphabetical order).

In case of weblogic server we have observed that it is creating servlet object based on the order which we have specified in development descriptor. We are used `load.on.startup` with the negative value as shown below.

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

When we deploy the above project we have observed that server is not creating servlet object. We have observed that same servers are removing the old servlet object and creating new servlet object when ever the .class file is modified and send the request to server.

We have observed this behavior in weblogic server. In the Tomcat server even if we modify the .class file it is not pickingup the lastest .class file until and unless we restart the server.

Conclusion: According to SunMicro systems servlet specification it's all the responsibility of server to create servlet object when ever it is needed. All the above scenarios we have observed in different servers.

When we are developing a project we follow the phases of software life cycle. That is requirements, analysis, design, coding and testing. In a project we will carryout multiple types of testing they are:

1. Unit Testing
2. Integretion Testing
3. System Testing (or) QA Testing
4. UAT Testing (User Acceptence Testing)

It is the responsibility of developer to perfrom Unit Testing and Integration Testing. The tester performs system level testing. It is the responsibility of end user to perform User Acceptence Testing. In a project we maintain different type of servers. They are:

1. Development Server
2. Integretion Server
3. QA server (Quality Assurance)
4. Production server

A development computer is used by a developer to develop the code and to perform unit testing. Integration server is used to perform only Integration Testing. QA server is used by the Tester to perform system level testing.

Production server is used by the client for the final deployments of the project as well as to perform UAT testing.

How do we deploey the project in Integration QA production servers?

To release web based applications we create war files and deploy the project. Procedure to create a war file and deploy the project.

1. Goto a folder where the project is available(C:\work\webapps).
2. Use the following command to create a war file.
`>jar-cvf lic.war`
3. Copy pastes the war file inside server specific folder.

When we deploy a war file server creates a folder with the war file name and copy the contents of war file into that folder.

Approach 3: We can deploy the projects through server admin console. Every server will have an admin console through which we deploy the projects. In case of Tomcat Server we have an option WAR file and deploy the project.

In case of weblogic server as part of the domain structure we have an option to deploy to the project. In weblogic server also upload the WAR file and deploy the project.

There are 3 methods available in servlet which act as servlet life cycle methods. They are:

1. init()
2. service()
3. destroy()

Generally we write any code which has to be executed only once when the servlet object is getting created inside init(). As part of the destroy() method we will provide the code which has to be executed only once at the time of removing the servlet object.

As part of the service() we provide the code which has to be executed repeatedly. The following is an example of using init(), service() and destroy() to get the connection only once and use it.

```
public class RetrieveRecordsServlet Extends HttpServlet{  
    Connection con;  
    public void init(ServletConfig config){  
        try{  
            DriverManager;  
            con.DriverManager;  
            System.out.println("connection got");  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
    public void destroy(){  
        try{  
            con.close();  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
    public void Service( ----- ){  
        try{
```

```

Statement stmt = con;
ResultSet rs = stmt.executeQuery("select * from emp");
printWriter
while(rs.next()){
out.println(rs.getString(1));
out.println(rs.getString(2));
out.println(rs.getString(3)+"<br>");
}
}catch(Exception e){
e.printStackTrace();
}
}
}

```

It's dangerous to use instance variable in servlets. It's always recommended to remove instance variables from servlets. It's recommended to use only local variables.

When we use instance variables in servlet we encounter lot of process. If multiple clients send the request to at the same time we are trying to develop the following servlet with out variable as instance variable.

```

public class TestServlet Extends HttpServlet{
PrintWriter out;
public void service( ----- ){
out = response.getWriter();
out.Println("First Line<br>");
try{
System.out.println("sleeping");
Thread.sleep(10000);
System.out.println("okup");
}catch(Exception e){
}
out.println("Second Line");
}
}

```

When we test the above servlet by sending multiple requests at the same time from different browsers we have observed unexpected outputs in the browsers. It's always

recommended to avoid using instance variables in servlet. It's always recommended to use only local variables in servlets.

Can we call destroy() from service()? // yes we can call.

When we call the destroy() from the service() every time server executes service() it will call the destroy(). Because of this servlet will not remove servlet object.

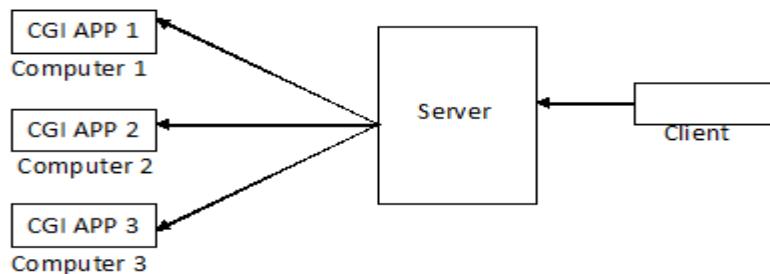
```
public class TestServlet implements Servlet{
```

```
-----  
Public void service(ServletRequest request, ServletResponse response){  
System.out.println("we are in service()");  
Destroy();  
}  
}
```

Before sun micro system introduce servlets there is a technology CGI(Common Gate way Interface). To develop web based applications.

There are so many draw backs are available in CGI. They are:

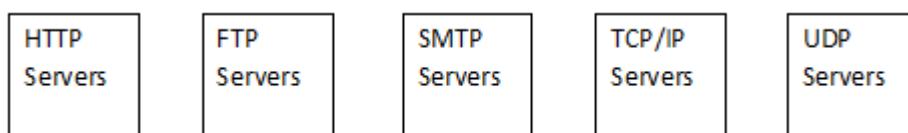
1. CGI applications are platform dependent.
2. The major disadvantage of CGI applications is they run in the computer which are associated with servers. Because of these reasons we find lot of performance issues.



In case of CGI applications when the client send the request server starts a new process. This process runs the CGI applications separately.

Sun micro system has introduced servlet API to resolve all the problems of CGI applications. There are so many protocols available in the market. Some of them are HTTP, FTP, SMTP, TCP/IP ,UDP and etc.

Based on these protocols there are so many servers are develop.



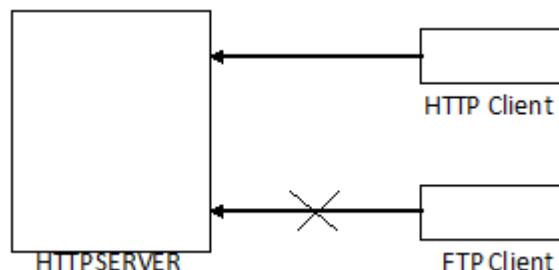
****What is HTTP server?**

A Server which are built based on HTTP protocol are called as HTTP Server.

Tomcat, weblogic, websphare, JBOSS, ----- are called as HTTP Servers.

The initial idea of sun micro system is to run the servlets on all the servers. That is HTTP servers, FTP servers, SMTP servers, TCP/IP servers and UDP servers etc.

An FTP client can't communicate with HTTP servers.



Sun micro system want to release multiple packages to support different protocols.

They are:

Java.servlet.http (all the classes and interfaces belongs to http protocol).

Java.servlet.ftp

Java.servlet.smtp

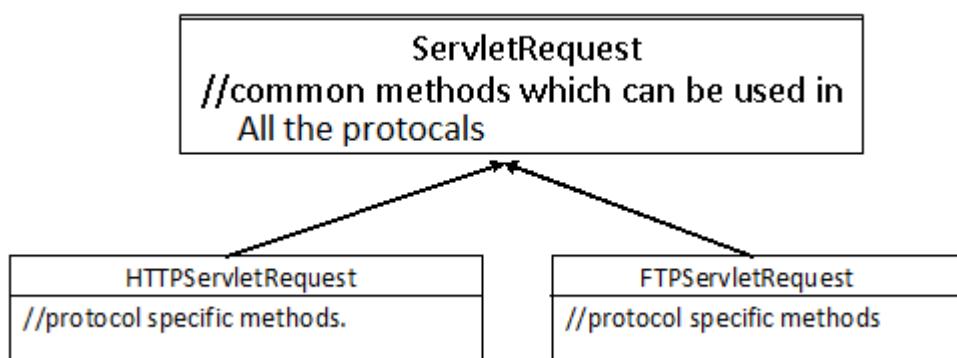
Java.servlet.tcp/ip

Java.servlet.udp

----- etc.

All the common functionalities between all the protocols is placed in a package javax.servlet package.

We call all the classes and interfaces which are part of javax.servlet as protocol independent methods.



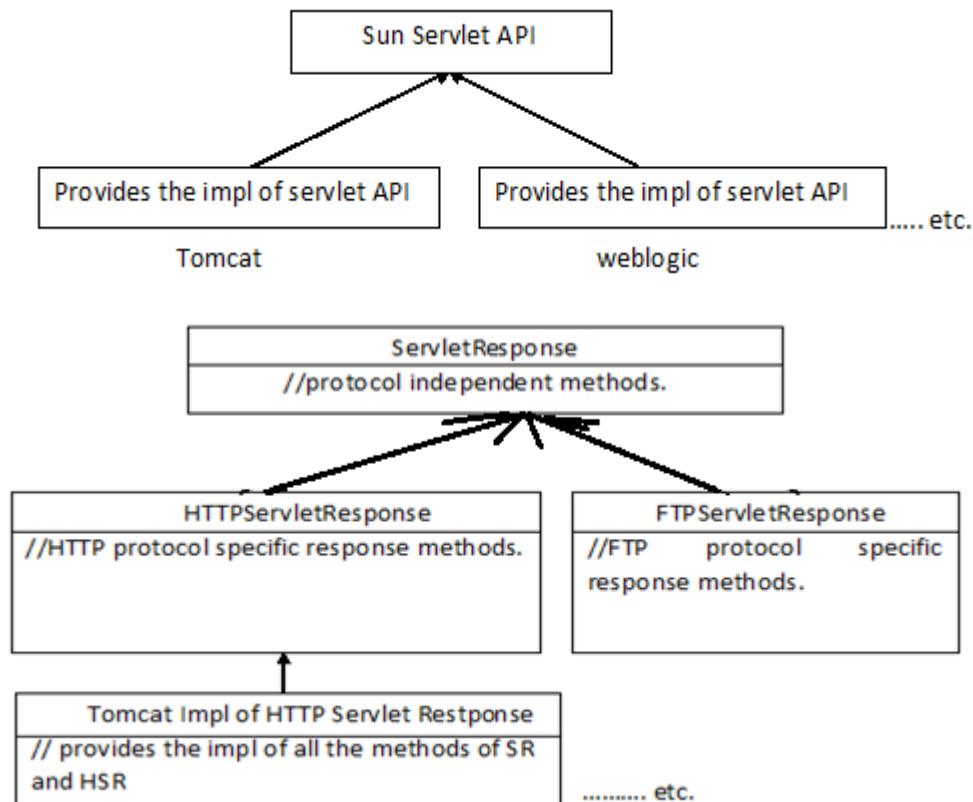
..... etc.

***Can we run the servlets on any server?**

We can run the servlets on any server, if they provide the implementation to servlet.API. If the server not provide the implementation to servlet.API ,we can't run the servlets on that server .

We can't run the servlets on HTTP web server, IIS server. This is because this server doesn't provide the implementation to servlet API.

If we want to run the servlets on any server, they have to provide the implementation to servlet API.



The meaning of creating request object and response objects are , creating an object to a class which provides the implementation of **HTTPServletResponse and **HttpServletRequest**.**

The names of these classes are varied from server to servers. Theoretically speaks we can deploy the servlets on any HTTPServers.

As there are there ways to develop the servlet which is the best way either servlet or generic servlet or **HTTPServlet**. It's not at all recommended to develop the servlet based on servlet interface. This is because we have to provide lot of redundant code in the project.

The following is the internal code of generic servlet class.

```
public abstract class GenericServlet implements Servlet{  
    ServletConfig config;  
    Public void init(ServletConfig config){  
        this.config = config;  
    }  
    public void destroy(){
```

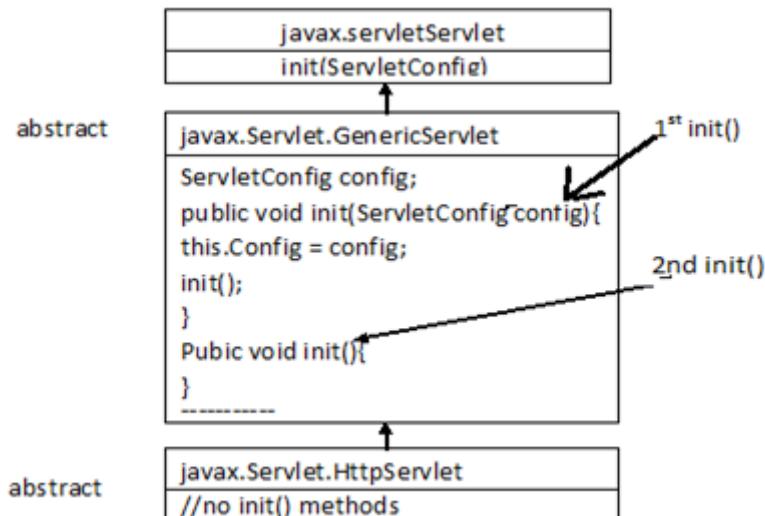
```
config = null;
```

```
-----  
}
```

The following Servlet is implements based on generic servlet to find method name sent by the client, url and useragent header value.

```
public class TestServlet Extends GenericServlet{  
    public void service(ServletRequest request, ServletResponse response){  
        HttpServletRequest req = (HttpServletRequest)request;  
        HttpServletResponse res = (HttpServletResponse)response;  
        PrintWriter out = response.getWriter();  
        out.println(request.getMethod());  
        out.println(request.getRequestURL());  
        out.println(request.getProtocol());  
        out.println(request.getHeader("user Agent"));  
        out.println(request.getHeader("Accept-Language"));  
    }  
}
```

The following diagram shows the over all picture of init() methods in servlet interface, GenericServlet and HttpServlet.



when we develop the servlets based on HttpServlet and if we would like to carry out and work only one time we have to use init().

It's always recommended to use/override second init() as part of servlets.

The following example demonstrates the use of 2nd init().

```
public class TestServlet Extends HttpServlet{  
    public void init(){  
        System.out.println("we are in 2nd init()");  
    }  
}
```

```

}
public void service(HTTSPortletRequest request, HTTSPortletResponse response){
    }
}

```

When we deploy the above project, Server is creating TestServlet object and then server calls 1st init() method. Now the server opens TestServlet object and check whether the 1st init() is available or not. If it is available it will Execute, if it is not available in TestServlet, server check in HttpServlet, if it available execute it, if not available, server goto GenericServlet class and check for 1st init() method. As the 1st init() is available in GenericServlet it will initialize instance variable config and it will call 2nd init() method. Now the JVM open TestServlet class and check for 2nd init() method. As it is available and get Executed. It's not recommended to use 1st init() as part of servlets which uses HttpServlet.

```

public class TestServlet Extends HttpServlet{
    public void init(ServletConfig config){
        System.out.println("we are in 1st init() method");
        System.out.println("doing some work for one time");
    }
    public void service(---- ){
        System.out.println("we are in service() method");
        ServletConfig config = getServletConfig();
        System.out.println(config);
        String drr = config.getInitParameter("drr");
        System.out.println(drr);
    }
}

```

When we run the above servlet program we got the NullPointerException this is because of the following reasons.

1. When the server has created TestServlet object it has initialize instance variable config of GenericServlet to Null value.
2. The server has called 1st init() method server has Executed 1st init() method of TestServlet. It has not executed 1st init() method of GenericServlet. Because of this reason it has not initialized config object of GenericServlet.
3. Because of this reason when we call getServletConfig() method returns Null value.

4. The best solution for the above problem is use 2nd init() method. Instead of using 2nd init() in the above TestServlet we have used 1st init() with an instance variable ServletConfig.

```
public class TestServlet Extends HttpServlet{
    ServletConfig config;
    public void init(ServletConfig config){
        this.config = config;
    }
    public void service(-----){
        System.out.println(config);
    }
}
```

The above code resolves the problem. But we have found lot of redundant code. Instead of using the 2nd init() method by using super.init(config).

```
public class TestServlet Extends HttpServlet{
    public void init(ServletConfig config){
        super.init(config);
    }
    -----
}
```

Generally in a project all the project information is maintain in an xml file. As part of the projects when ever we deploy the project we should be able to create the object to the servlet and read the contents from configuration file and store it in JVM's memory.

When ever we url deploy the project we should be able to remove the contents from JVM's memory. The contents which is storing JVM's memory should be able to accessable in service(). Our project should be able to read the configuration when ever we deploy the project.

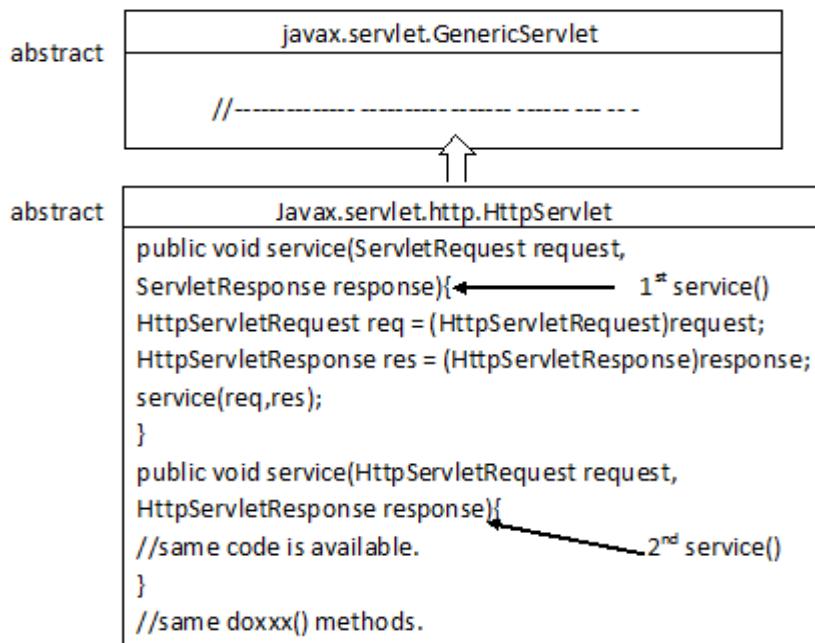
```
public class ActionServlet Extends HttpServlet{
    public void init(){
        ServletConfig config = getServletConfig();
        String filename = config.getInitParameter("config");
        System.out.println("Reading the contents from" +filename+ "and store in JVM");
    }
    public void service(HttpServletRequest request, HttpServletResponse response){
        System.out.println("using the contents");
    }
    public void destroy(){
        System.out.println("Remove the contents");
    }
}
```

```
    }  
}
```

To achieve our requirement we have use the following information in web.xml file.

```
<web-app>  
    <servlet>  
        <servlet-name>action</servlet-name>  
        <servlet-class>ActionServlet</servlet-class>  
        <init-param>  
            <param-name>config</param-name>  
            <param-value>/WEB-INF/lms.config.xml</param-value>  
        </init-param>  
        <load-on-startup>2</load-on-startup>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>action</servlet-name>  
        <url-param>*.do</url-param>  
    </servlet-mapping>  
</web-app>
```

The following is the same piece of code available as part of HttpServlet.



It's always recommended to develop the servlets based on 2nd service() method. The advantage of this approach is we no need to typecast ServletRequest, ServletResponse into HttpServletRequest and HttpServletResponse.

Ex: public class TestServlet Extends HttpServlet{

```

public void service(HttpServletRequest request, HttpServletResponse response){
    System.out.println("we are in 2nd service()");
}
}

```

The following steps are carried out by the server when the client send the Request.

1. Server always call the 1st service() method. Now the server open TestServlet and check whether 1st service() is available or not. If it is not available it will goto HttpServlet and Execute the 1st service().
2. The internal code of 1st service() method typecast ServletRequest, ServletResponse into HttpServletRequest and HttpServletResponse and call 2nd service() method is available or not. If it is available it will execute.

While we are developing the project some times we should be able to process only either post request or get request. In this scenario we override doGet() or doPost() methods.

It is the responsibility of HttpServlet 2nd service() to call appropriate doxxx().

Ex:

```

public class LogicServlet Extends HttpServlet{
    public void doGet(HttpServletRequest request, HttpServletResponse response){
        PrintWriter out = response.getWriter();
        System.out.println("can't process your request");
    }
    public void doPost(HttpServletRequest request, HttpServletResponse response){
        PrintWriter out = response.getWriter();
        System.out.println(request.getParameter("uname"));
        System.out.println(request.getParameter("pwd"));
    }
}

```

When do we use 2nd service() or doGet() or doPost() methods?

If the application has to move for all the requests then we need to over ride 2nd service(). If the application has to work only for a specific method then we use either doGet() or doPost().

In a project some times we want to use doGet() and doPost() methods instead of 2nd service() and we would like to remove redundant code.

Ex:

```

public class LogicServlet Extends HttpServlet{
    public void doGet(HttpServletRequest request, HttpServletResponse response){

```

```

        process(request,response);
    }
    public void doPost(HttpServletRequest ----){
        process(request,response);
    }
    public void process(HttpServletRequest request, HttpServletResponse response){
        PrintWriter out = response.getWriter();
        out.println("from process() <br>");
        out.println(request.getParameter("uname"));
        out.println(request.getParameter("pwd"));
    }
}

```

We can say there are 'n' number of ways are there to develop the servlets.

Ex: some body has developed a class which inherits the properties of HttpServlet and make the class as abstract.

```
public abstract class InetsolvServlet Extends HttpServlet{
}
```

Any body can develop the servlets based on the above InetsolvServlet. Because of this reason we say there are 'n' number of ways are there to develop the servlet. Generally in every project we will try to develop our own servlets. These servlets we will try to use our own methods.

Ex:

```
public class abstract BMSServlet Extends HttpServlet{
    public void init(){
        doOneTimeWork();
    }
    public void doOneTimeWork() { }
    public void service(HttpServletRequest request, HttpServletResponse response){
        performWork(request,response);
    }
    abstract public void performWork(HttpServletRequest request,
        HttpServletResponse response) ;
}
```

The following is the FirstServlet Example based on BMSServlet.

```
public class FirstServlet Extends BMSServlet{
    public void performWork(HttpServletRequest request, HttpServletResponse response){
        PrintWriter out = response.getWriter();

```

```
    out.println("we are in performwork()");  
}  
}
```

Most of the times we develop the servlet classes with package names while we are configuring the servlet in web.xml file which contain the packages we must specify absolute class name.

```
//>javac -d .FirstServlet.java  
<servlet>  
    < servlet-name>fs</servlet-name>  
    < servlet-class>info.inetsolv.FirstServlet</servlet-class>  
</servlet>  
//pdf.coreservlet.com
```

Once if a develop a project for http protocol we can deploy the some project on https protocol also. We need to enable https option in the server (SSL option).

*****JSP*****

JSP stands for java server pages and it **is use to develop web based applications**.

1. Generally we place all the JSP programs inside project folder. Every JSP program ends with an extension ".JSP ". We no need to configure JSP as part of web.xml file. We can send the request to the JSP's directly from the browser.

Ex: <http://localhost:8000/webapp/one.jsp>

2. There are so many advantages are available with JSP's when compared with Servlets. The following are some of the advantages of JSP's when compared with servlets.

2.1. JSP's improve the productivity of the developers (we can develop the JSP projects quickly when compared with servlets).

Thearitically : We can develop JSP's without knowledge of java.

Practically : To develop JSP's we need knowledge of java.

2.3. By using JSP's we can separate business logic and presentation logic.

3. When we deploy the project and client has send the request to JSP **the server has converted JSP program into corresponding servlet.**

4. An Apache guy has released HttpServer. This server can process HTML pages, images and JavaScript. This server can't run servlet's and JSP's.

***What is a Servlet Container?**

Servlet container is a java program which provides the implementation of Servlet-API. It is the responsibility of Servlet Container to load the .class file and create Servlet object and call the life cycle methods of servlet.

***What is JSP container?**

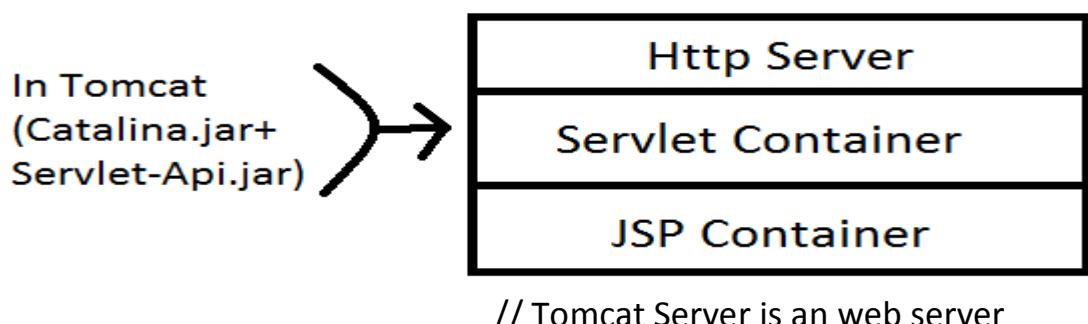
A JSP container is a java program which provides the implementation of JSP-API. It is the responsibility of JSP container to run JSP programs. And convert JSP into corresponding Servlets .

Every JSP Container having JSP Compiler , JSP Compiler is a JAVA Program.

Jsp compiler is available in weblogic.jar (in Weblogic) , in jsaper.jar (in Tomcat).

A Tomcat Server is a collection of HttpServer, Servlet Container and JSP container.

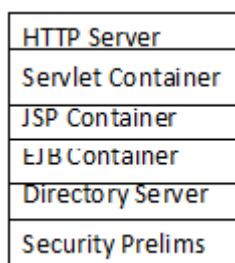
So Tomcat Server called as Web server.



Today's servers are aggregated into two ways. They are:

1. Web Servers
2. Application Servers

Most of the web servers run servlets and JSP's. Application servers run servlets and JSP's as well as they can run EJB applications also.



Web logic Server is an Application Server

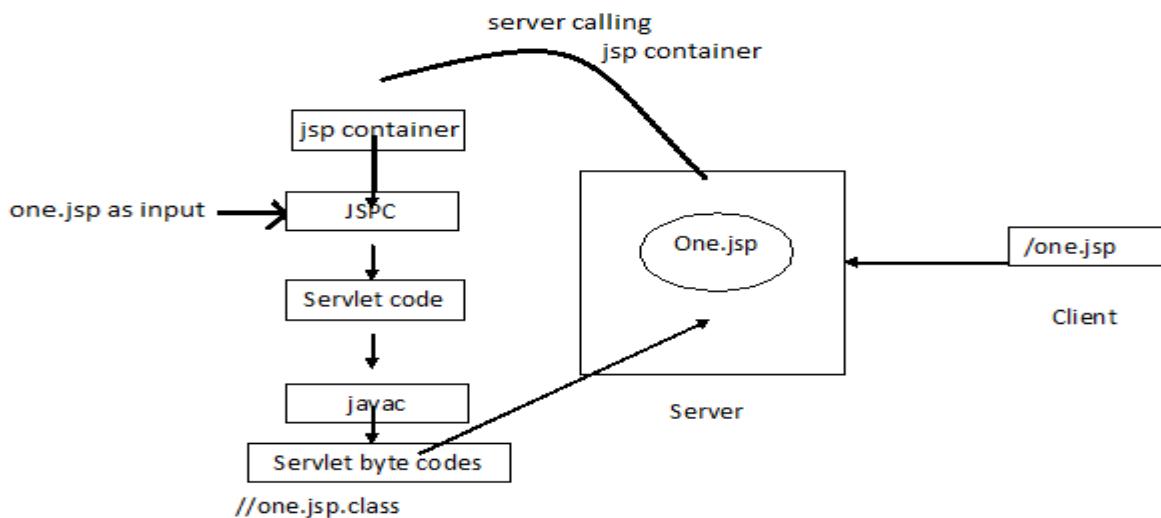
Tomcat is called as Web Server.

Web logic, Web Sphere, JBoss are called as Application Servers.

An application server provides the complete implementation to JEE.API. Where as web servers can provide the partial implementation of JEE.API.

When we develop the JSP programs it is the responsibility of JSP compiler to convert JSP file into its corresponding Servlet program.

The following steps will be carried out when a client sends a request to server.



The server will carry out all the above steps when the client has sent the request to the server for the 1st time. When the client has sent the request 2nd time directly server execute the servlet object of the corresponding JSP.

Sun micro systems have inherited so many features of JSP programming language and develop JSP applications. Theoretically speaking we can develop JSP applications without writing Java class.

According to the discussion as of now we have observed that we can develop the projects quickly and deliver to the customer easily based on JSP's.

No doubt **servlets gives better performance when compared with JSP's.**

Even though servlet's give better performance when compared with JSP's. We can't measure the performance difference with human being eye. But when we consider the project cost we had a huge difference between servlet's and JSP's. In most of the projects we use both servlets and JSP's. **If you want to execute huge amount to business logic we use servlets. In the remaining scenario's we use JSP's.**

It is the responsibility of the server to call the JSP compiler when the client sends the request to JSP. We can call JSP compiler manually. Generally we do this work to understand the internal code of generated servlet.

Procedure to call web logic JSP compiler:

1. Set the class path to multiple jar files by using "setDomainEnv.cmd" (JSPEC program available inside web logic.jar).
2. Use the following command to generate the servlet.

```
>java weblogic.jspc -keepgenerated one.jsp
```

To place the source code(.java file)

By default web-logic JSP compiler removes .java programs. To retain the .java programs we use an option –keepgenerated.

The following are the rules of JSP compiler to follows set of rules in converting JSP into corresponding Servlet program.

1. Every JSP compiler must generate a servlet and place it in .java program. Because of the above rule different servers are following different naming conversions in writing the servlet name (.java file).

For Web-logic they are using the file name like _One.java

For Tomcat they are using One_jsp.java

2. Every generated servlet must be place inside a package. Different servers used different package names. (org.apache.jsp in Tomcat , jsp_servlet in Weblogic).

3. Every generated servlet must import 3 different packages.

3.1. import javax.servlet.*;

3.2. import javax.servlet.http.*;

3.3. import javax.servlet.jsp.*;

4. According to JSP specification every JSP compiler has to develop their own servlet. In that servlet we must override 2nd init() and 2nd service() method. 2nd service() method must be call _JSPService() method.

The following is the internal code of Tomcat JSP compiler.

```
public abstract class HTTPJSPBase extends HttpServlet{  
    public void init(){ }  
    public void service(HttpServletRequest request, HttpServletResponse response){  
        _JSPService(request, response);  
    }  
    Abstract public void _JSPService(HttpServletRequest request,  
                                     HttpServletResponse response);  
}
```

5. When the JSP compilers develop a Servlet for the corresponding JSP, we must declare the class as final.Because of nobody can inherit the properties of that class.

6. According to JSP specifications the parameters of `_JSPService()` method must be request and response only.

The following is the generated code of Tomcat JSP compiler.

```
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
public final class One_JSP extends HttpJspBase{
    public void _JspService(HttpServletRequest request,
                           HttpServletResponse response) {
        // standard local variables
    }
}
```

When we observed the generated servlet of Tomcat and web-logic we have found extra methods. These methods are specific to the servers. These methods are not declare as part of JSP specification.

When the client send the request to server, server executes `_JSPService()` method.

In windows-7 OS if we are not able to see the generated servlet then we need to disable set of JSP elements. **A JSP program contains set of JSP elements**. The following are the elements of JSP. They are given below:

1. Template Text
2. Script let
3. JSP Expressions
4. JSP Declarations
5. JSP Directions
6. JSP Action Tags
7. JSP Custom Tags
8. EL Expressions(Expression Language)

Template Text: This Element is used to send output to the client. We can write the Template Text Element directly in the JSP without any special symbols.

Ex: First Line //TemplateText Line1
 Second Line //TemplateText Line2

When we use Template Text the content will be placed inside `out.print("")` method.
When we run the above JSP we got the following servlet.

```
public class Final one_jsp extends HttpJSPBase{
    public void _JSPService(.....){
```

```

        out.Println("First Line");
        out.Println("Second Line");
    }
}

```

The following is an example of sending a form to the client by using JSP's.

```

<form> User Name:<input type = "text" name = "uname"/><br>
        Password:<input type = "Password" name = "pwd"/><br>
        <input type = "submit" value = "Login"/>
</form>           //One.jsp

```

Scriptlets: Scriptlets are used to write java code in JSP's.

Syntax: <% /start scriptlet

Java code %> //ending script

Ex: <% int a = 10;
 System.out.println(a);
%> //One.JSP

When the JSP complier encounter a Scriptlet directly placed inside `_JSPService()` method. When we are using Scriptlets in the JSP. We must follow all the rules in the java. If we don't follow all the rules of java the JSP compiler **will fail in converting .java program into .class file** and JSP compiler displays Errors in Browser.

Implicit variables:

In JSP's we can use some variables without we declaring it, we call them as implicit variables or implicit objects. The following are of implicit variables in the JSP.

1. request 2) response 3) pageContext 4) session 5) application 6) config 7) out 8) page 9) Exception

Response: Response implicit variable can be used directly in the JSP script without we declaring it. following is an example of using response object directly in the JSP.

```
<% response.setContentType("text/xml"); %>
```

We can send the error message directly to the client.

```
<% response.sendError(543,"abc"); %>
```

Request: Request implicit variable can be used directly in the JSP script without we declaring it.

request -----→ HttpServletRequest

```
<% out.println(request.getMethod()+"<br>");
        out.println(request.getRequestURI()+"<br>");
        out.println(request.getProtocol()+"<br>"); %>
```

Req: Develop the following form based application by using JSP's.

User Name:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="Login"/>	

The following details are you have Entered.
User Name: system
Password: malli

```
<%      String uname = request.getParameter(uname);
        String pwd = request.getParameter(pwd);    %>
```

The following details are you have entered

User Name: <% out.println(uname); %>

Password: <% out.println(pwd); %>

Out: We can use out variable directly in the JSP scriplet without we declaring it.

out -----→JSPWriter

JSPWriter is an abstract class which is available in javax.servlet.jsp package.

So we can't create object to JSPWriter class , server creates an object to a class which inherits the properties of JSPWriter class. In servlet we are used printWriter to send the output to the client. In JSP we use JSPWriter.

*What is the difference between PrintWriter and JSPWriter?

Every JSPWriter is associated with 8KB of internal Buffer. Where as PrintWriter doesn't associated with any Buffer.

Ex: We can use both print and write() is an out variable.

```
<%    int a = 10;
        int b = 20;
        out.print("A value is:" +a);
        out.write("B value is:" +b);  %>
```

XML: xml is platform independent, when writing the xml file we must follow all the rules of dtd file given by Sun micro system. We can read the content of xml file if it is a valid xml file. If it is invalid xml file we can't read the content from xml.

1. If we want to develop a xml file we required either dtd file or xml sequence file. These documents contains all the tags which has to be used by xml developer.
2. To resolve the xml problems in servlets sun micro systems has given a dtd file. This is used by all the server developers as well as servlet developers to write xml file.

3. Sun micro systems has released servlets dtd document. All the servers have used this dtd and develop parser program. As developers we can follow the dtd file and create web.xml file.
4. Because of security reasons we can configure JSP's into web.xml file.

Whenever we write xml file based on dtd document we must specify in web.xml file.

```
<?xml version = "1.0"?>
```

```
<!DOCTYPE web-app PUBLIC "web.xml" "http://java.sun.com/dtd/web-app\_2\_3.dtd">
<web-app>
</web-app>      //web.xml
```

To resolve all the security related problems in JSP's we can configure the JSP in web.xml file. The following is the configuration we use to configure a JSP into web.xml file.

```
<web-app>
  <servlet>
    <servlet-name>a</servlet-name>
    <jsp-file>/WEB-INF/one.jsp</jst-file>
  <servlet>
  <servlet-mapping>
    <servlet-name>a</servlet-name>
    <url-pattern>/a</url-pattern>
  <servlet-mapping>
</web-app>
```

Note: when we are writing <jsp-file> the path of the JSP must start with "/" .

1. If we don't want to allow any body to access the JSP directly recommended to place inside WEB-INF folder.
2. We can use <load-on-startup> as part of web.xml file for a jsp.

```
<servlet>
  <servlet-name>a</servlet-name>
  <jsp-file>/WEB-INF/jsp/one.jsp</jst-file>
  <load-on-startup>4</load-on-startup>
</servlet>
```

3. To a JSP when we use load-on-startup and when we deploy the project immediately server will convert jsp into corresponding servlet and the servlet.class will be loaded into JVM's memory and it creates the servlet object.

Config: Config variable is belongs to servletConfig data type. By using servletConfig we can remove the hard coding.

Config----→ servletConfig

```
<%    String drv = config.getInitParameter("drv");
      out.print(drv);  %>
```

servletConfig is an object used by servlet container to pass the information to a servlet during initialization.

Once if we get the ServletConfig object, we can find the name of the servlet which is configured in web.xml file by using a method getServletName() method.

Ex:

```
<%
    String name = config.getServletName();
    oput.println(name);
%>
```

As part of the url pattern if it end with .jsp, jsp compiler directly processes the jsp program.

Application: application ----→ ServletContext

1. Application implicit variable can be used directly in the jsp.
2. ServletContext object contains set of methods which are used to interact with ServletContainer. By using these methods we can find the information about ServletContainer.
3. When ever sun micro system releases a new version of API. It is the responsibility of server vendor to provide the implementation to the latest API. For example when servlet 2.5 is released Tomcat guys has provided the implementation and integrated it in Tomcat 6.0 the latest version of servlet API is 3.0 this version is integrated in Tomcat 7.0.

*What is the different between servlet-API 2.5 and servlet-API 3.0?

The servlet-API 3.0 supports processing annotations.

Session:

We can use session implicit variable directly in the jsp. Session implicit variable is holding Http Session object.

Session ----→ HttpSession

Page:

We can use page implicit variable directly in the JSP.

Page ----→ Object

Page implicit variable hold the currently executes Servlet object for the corresponding JSP.

Exception:

The implicit variable can be used only in Error pages. When we try to use this variable in a JSP which is not an Error page we get an Error message.

Exception -----→ Throwable

PageContext:

We can use PageContext object directly in the jsp.

PageContext ---→ PageContext

1. Whenever `_jspService()` method is Executing it creates PageContext object. Whenever `_jspService()` method Execution is completed it removes PageContext object.
2. By using PageContext implicit variable we can get any other implicit variable object.

Ex: `application = PageContext.getServletContext();`

In the jsp program also we can use comments.

- i. We can follow HTML style of comments in jsp.

Ex: `<!--`

```
<%
    out.println("we are in scriptlet");
%
--- >
```

- ii. jsp style of writing comments.

```
<% -->
```

```
    code
```

```
----%>
```

JSP Declarations:

JSP declarations are used to create instance variables, instance methods and static variables and static methods.

Syntax:

```
<%!
```

```
    // instance variables
    // static variables
    // instance methods
    // static methods
```

```
%>
```

When we run the JSP program which is having JSP declarations the JSP compiler directly place the contents inside the class.

Ex:

```
<%!
    int a;
    public void method(){
    }
%>          // One.jsp (JSPC)
public final class one_jsp Extends HttpJSPBase{
int a;
public void method(){
}
public void _JSPService(-----){
// implicit variables      }
}          // One.servlet
```

If we write the JSP declaration with set of methods we need to call the methods from _JSPService() method.

Ex:

```
<%
    public void methodOne(){
        System.out.println("we are in methodOne()");
    }
%>
<%
    methodOne();
%>
```

The implicit variables of JSP can't be accusable in JSP declarations.

Ex:

```
<%
    Public void methodOne(){
        out.println("we are in methodOne()");
    }
%>
```

According to sun micro system the implicit variables can't be accusable in JSP declarations. This is because the local variables of one() method can't be accusable in another method.

As a java programmer we can follow our own techniques to use the local variables of one() method in another method.

Ex:

```
<%!
    JSPWriter out;
    public void methodOne(){
        out.println("we are in methodOne()");
    }
%>
<%
    this.out = out;
    methodOne();
%>
```

The following are three life cycle methods of JSP.

1. JSPIInit()
2. _jspService()
3. jspDestroy()

Ex:

```
<%
    public void JSPIInit(){
        System.out.println("we are in JSPIInit()");
    }
    public void jspDestroy(){
        System.out.println("we are in jspDestroy()");
    }
%>
<%
    System.out.println("we are in _jspService()");
%>
```

When the corresponding servlet object for the jsp is created, server call JSPIInit() method. Every time the client send the request it Execute _jspService() method. When the project is un-deploy server Executes jspDestroy() method.

In case of jsp when ever we modify the jsp and send the request. The Existing servlet object will be removed and create new servlet object.

When the server create servlet object for a corresponding jsp program?

Scenario:

1. When the client send the request to jsp for the 1st time server create the 1st servlet object and call the life cycle method.
2. When we modify the jsp and when the client send the request to the server if already servlet object is available it removes the old servlet object and creates new servlet object [it will call `jspDestroy()`].

JSP Expressions: jsp Expressions simplifies the use of java Expressions in jsp.

Getting the value from variable and display to the client is called as java Expressions.

Ex:

```
<%  
    int a = 10;  
    out.println(a);  
%>
```

Performing arithmetic, relational, logical operations on a variable one also called as java Expression.

Ex:

```
<%  
    int a = 10;  
    int b = 20;  
    out.println(a+b);  
%>
```

In a project we use some supporting classes every supporting class must be placed inside a package other wise server will not recognize the supporting classes.

Displaying an object to the client is also considered as java Expressions.

Calling the methods an object and send the output in the client is also called as java Expressions.

Ex:

```
<%  
    Infodreamsoft.Emp e = new infodreamsoft.Emp();  
    out.println(e);    or    (e.toString());    or    (e.Eno());  
%>
```

Syntax:

```
<% = javaExpressions %>
```

The following is an Example of jsp Expressions.

```
<%  
    int a = 10;
```

```

int b = 20;
%>
<% = a + b %>      or      <% = a%>

```

The following is the code how jsp compiler convert jsp Expressions.

```
<% = a + b%>
```

JSPC
↓

```
Out.println(a + b);
```

```
<%
```

```

info.inetsolv.Emp e = new info.inetsolv.Emp();
e.setEno("1");
e.setName("Raju");

```

```
%>
```

```
<% = e%>
```

```
<% = e.getEno() %><br>
```

The following jsp get the data from the form and display to the client.

Eno	<input type="text"/>
Name	<input type="text"/>
Salary	<input type="text"/>

The following details
 Eno....
 Ename....
 Salary....

Eno:<% = request.getParameter("Eno")%>

Ename:<% = request.getParameter("Ename")%>

JSP Directive:

JSP directive is an instruction given to JSP compiler. Based on the instruction which we are given the JSP compiler converts JSP code into corresponding java code. There are 3 directives are available. They are:

1. page directive
2. include directive
3. taglib directives

syn:

```
<%@ directive attribute1 = " " attribute2 = " "%>
```

The following are the attributes of page directive.

- 1) Language
- 2) import
- 3) info
- 4) contentType
- 5) buffer
- 6) autoFlash
- 7) isErrorPage
- 8) errorPage
- 9) session
- 10) extends
- 11) isThreadsSafe

Without knowledge of java it is very difficult to develop servlets. Sun micro system has derived so many concepts of ASP and developed JSP. We can use JSP's to develop web-based applications without writing java code in it.

Note: According to sun micro system we can use any scripting language in the JSP to develop web-based application.

In the JSP's we can use java code also According to sun micro system I can provide a java code in a script or VB Script in a scriplet.

Language page directive:

This directive is used to specify the language is provides. According to JSP specification we can provide any scripting language as part of scriplet to specify the scripting language. They are using the language page directive as shown below.

```
<%@ page language = "javascript"%>  
<%    var v = 10;  %>
```

The servers like tomcat web logic or not supporting any scripting language a part from java. A server Resin has supported a java and java script as part of scriplet. The latest versions of Resin also are not supporting java script as part of scriplet.

If we doesn't provide language page directive by default JSP compiler treats it as java code is provide. We can provide the directive any where in the JSP.

import page directive: import page directive is used to import a package in generated servlet.

Ex:

```
<%@ page import = "java.util.*"%>  
<% ArrayList list = new ArrayList();  
    list.add("one");  
    list.add("two");  
%>
```

We can import multiple packages by using single import page directive.

```
<%@ page import = "java.util.* , java.sql.*"%>
```

To import multiple imports we are using is a separator. It is not recommended to import multiple imports by using import page directive. Because of this version we will get maintained problem.

```
<%@ page import = "java.util.*"%>  
<%@ page import = "java.sql.*"%>
```

Oracle

Database: It is collection of meaningful data.

Ex:

<u>SNO</u>	<u>SNAME</u>	<u>MARKS</u>
101	MALLI	99
102	SREE	98
103	KRISH	97
104	VISH	96

Management System: It is a software it helps to manage the database management system should able to perform the following activities very easily.

1. Inserting the new data.
2. Updating the exiting data.
3. Deleting unnecessary data.
4. Retrieving the require data.

A database along with the software which helps to manage. The database is called database management system (DBMS).

A DBMS which is based on relational theory is called as relational database management system.

Examples of RDBMS:

1. ORACLE
2. SQL SERVER
3. DB2
4. MYSQL
5. SYBASE
6. TERA DATA
7. MS ACCESS

SQL:

Structured query language pronounced as (SEQUEL). This language is used to communicate to oracle database.

Features of SQL:

1. It is a command based language.
2. It is not case sensitive.
3. Every command should end with ‘;’.

4. Every command starts with “verb”.
5. It is similar to English. This language is developed in the year 1972. Mr.CODD, by IBM developed by “IBM”.

SUB LANGUAGE OF SQL:

1. DDL (Data Definition Language)
2. DML (Data Manipulation Language)
3. DRL/DQL (Retrieval/query)
4. TCL (Transaction Control Language)
5. DCL (Data Control Language)

***DDL**: This language is used to manage database objects. It is collection of five commands
CREATE, ALTER, DROP, TRUNCATE, RENAME

***DML**: This language is used to manipulate the data you have stored. It is collection of four commands. INSERT, UPDATE, DELETE, MERGE

***DRL**: This language is used to retrieve the data from the database. It is collection of only one command. SELECT

***TCL**: It is used to maintain the transaction of Oracle database. It is collection of three commands. COMMIT, ROLLBACK, SAVEPOINT

***DCL**: This language is used to control the access of the data to the users it is collection of two commands. GRANT, REVOKE

TABLE: Table is an object which is used to store some data. In general it is collection of Rows and Columns.

CREATE command: This command is used to create a table.

Syntax:

```
CREATE TABLE<TABLE_NAME>(COL_NAME1 DATATYPE(SIZE),COL_NAME2
DATATYPE(SIZE), COL_NAME3 DATATYPE(SIZE));
```

Ex: Create table student (SNO number (3), SNAME varchar2(20), MARKS number(3));

SQL * PLUS: it is a client environment. SQL>Conn: it is called connect. Default username: Scott Password: tiger
--

SQL*PLUS: It is a client environment where the developer will write the queries and submit the queries for execution.

***TO ESTABLISH CONNECTION:**

```
SQL>CONN  
USER NAME: SCOTT  
PASSWORD: TIGER  
CONNECTED
```

***INSERT command:** It is used to insert the values into the data base table.

Syntax: INSERT INTO <TABLE_NAME> VALUES(VAL1,VAL2,VAL3,.....VALn);

Ex: Insert into student values(101,Arun,60);

Error: Arun

Correct: 'Arun'

Insert into student values(101,'Arun',60);

Note: for varchar to value we need enclose with single(' ') codes.

Insert into student values(102,'kiran',86);

Insert into student values(104,'vijay');//this statement is wrong

Insert into student values(105,'vijay',null);

Note: Null is a keyword which is used to represent unavailable or undefined symbol.

Insert into student values(106,null,null);

***SELECT:** This command is used to return the data from the table.

Syntax1: SELECT * FROM <TABLE_NAME>;

Ex: select * from student; // * represent ALL

Note: where we use * all the information (ALL the columns and the rows are display).

Syntax2: for insert command:

INSERT INTO <TABLE_NAME> (COL1,COL2,....COLn) VALUES (VAL1,VAL2,.....VALn);

Ex: insert into student (SNO,SNAME) values (106,'Amit');

***insert the values at runtime using '&' operator:**

Ex: INSERT INTO STUDENT VALUES (&SNO,'&SNAME',&MARKS);

***SELECTING SPECIFIC COLUMNS:**

Ex: Select SNO, MARKS from student; Select MARKS, SNO from student;

<u>SNO</u>	<u>MARKS</u>	<u>MARKS</u>	<u>SNO</u>
101	60	60	101
102	85	85	102
----	----	----	----

---	---	---	---
108	98	98	---

Select SNO, SNAME from student;

Select * from student;

Select * From student; // not case sensitive

Create table employee(ENO number(2),ENAME varchar2(20),SALARY number(5),JOB varchar2(20));

Insert into employee values(101,'vijay',10000,'HR');

Insert into employee values(102,'ABC',33000,'GM');

*write the one column to be display in the student?

Select SNAME from student;

Select EMPNO,ENAME,SAL,JOB FROM EMP;

We can perform some calculation using Arithmetic operations to desired output.

Ex: Select EMPNO,ENAME,SAL,SAL*12,DEPTNO from EMP;

<u>EMPNO</u>	<u>ENAME</u>	<u>SAL</u>	<u>SAL*12</u>	<u>DEPTNO</u>
101	Mallikarjuna	50000	600000	GM
102	---	---	---	---

***Columns ALIAS:** Column ALIAS user defined column heading given to the required column.

Ex: Select EMPNO,ENAME,SAL,SAL*12 ANNUAL_SAL,DEPTNO from EMP;

Select EMPNO ROLL_NO,ENAME,SAL from EMP;

Note: 1. Column Alias can be provided for any column.

2. Column Aliases are temporary.

3. The scope of column Alias is to respect to the same query.

***DATA TYPES:**

1. **CHAR:** The data type is used to store alpha numeric values.

It can be also store special symbols like -,:,:,* and etc.

This data type of fixed size. The Maximum size is 255 bytes.

Note: Memory is not used efficiently.

2. **VARCHAR2:** This data type it can store alphanumeric values special symbols like -,:,_ and etc. This data type is of variable size. Maximum size 4000 bytes.

Note: Memory is efficiently used.

3. **NUMBER(P,S):** p-precision/s-scale

This data type used to store numeric values maximum size 38 digits.

4. **DATE**: This data type is used to store date values it will not have size.

Range 01-JAN-4712 BC TO 31-DEC-9999 AD

Default Date format is DD-MM-YY.

Ex: Create table student1(SNO number(2), SNAME varchar2(20),MARKS number(3), DOJ Date);

Insert into student1 values(101,'Ravi',99,SYSDATE);

5. **LONG**: Similar to varchar2 data type. Maximum size 2 GB.

6. **RAW**: used to store Images,logos,digital signatures etc. Maximum size 255 bytes.

7. **LONG RAW**: Similar to RAW data type. Maximum size 2 GB.

8. **CLOB**:(character large object) used to store characters. Maximum size 4 GB.

9. **BLO**:(binary large object) used to store binary data. Maximum size 4 GB.

10. **BFILE**:(binary file)

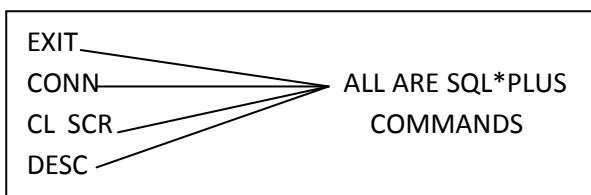
***Describe command**: This command is used to see the structure of the table.

Ex: DESCRIBE Student ///(not use ;)

<u>NAME</u>	<u>NULL</u>	<u>TYPE</u>
SNO		NUMBER(2)
SNAME		VARCHAR2(20)
MARKS		NUMBER(3)

NOTE: It is SQL*PLUS command

Short form is DESC



***WHERE clause**: class is used to select specific rows basing on a condition

Ex: Select * from emp where sal>2000;

SAL

2975

2875

3000

Select * from emp where deptno = 10;

Select * from student where sno < = 103;

Select eno,ename,sal,deptno from emp where dept = 20;

Note: Data which is stored the inside the table is case sensitive.

Ex: Select * from emp where job = 'clerk';

Select * from student where comm is NULL;

Note: is NULL is operator which is use to retrieve the rows based on NULL values.

Ex: Select * from student where marks is NULL;

Keyword: it is display distinct values (unique values).

Duplicates of suppressed.

Ex: select distinct deptno from emp;

DEPTNO

30

20

10

Select distinct job from emp;

UPDATE: The command is used to change data present in the table.

Syntax: Update<TABLE_NAME> set <COL_NAME> = <VALUE> where <CONDITION>;

Ex: Update student set marks = 30 where sno = 102;

Update emp set sal = 1000 where ename = 'scott';

Update emp set ename = 'Arun' where eno = 7369;

NOTE: when where clause is not use, all the rows are updated.

Ex: Update emp set sal = 5000;

Update emp set sal = 3500 where comm is NULL;

Update student set marks = NULL where sname = 'kiran';

Update emp set sal = NULL where comm is NULL;

Update emp set job = 'HR', sal = 5000 where ename = 'Alen';

***DELETE:** This command is used to remove the complete row from the table.

SYSTAX: Delete from <table_name> where <condition>;

EX: *write a query delete command?

Delete from student where sno = 101;

Update student set sname = null, marks = null where sno = 101;

Note: when where clause is not use all the rows delete.

Ex: Delete from student1;

***LOGICAL OPERATORS**: There are three logical operators. They are

1. AND
2. OR
3. NOT

AND(T AND T = T):

Syntax: select * from EMP where SAL > 2000 AND JOB = 'MANAGER';

Note: AND operator will return the rows when all the conditions are satisfied.

Select * from emp where SAL > 2000 AND JOB = 'MANAGER' AND deptno = 30;

Select * from emp where DEPTNO = 10 OR DEPTNO = 20;

Select * from emp where SAL > 2000 AND SAL < 4000;

Write a query to delete first and last rows?

Ex: Select * from empno = 7499 OR 7521 OR 7566/

Select * from empno = 7499 OR empno = 7521 OR empno = 7566;

Note: query to display the rows who are not managers?

Select * from emp where NOT job = 'MANAGER';

Select * from emp where NOT deptno = 30;

BETWEEN: between operator is used to display the rows which is following in the range of values.

Select * from EMP where SAL BETWEEN 2000 AND 3000;

Note: Extreme values are included

Always specify first lower limit first and then higher limit.

IN OPERATOR:

1. IN operator will be return the rows when the values are matching in the list.

2. IN operator can be use as a replacement of OR operator.

Ex: Select * from EMP where EMPNO IN(7369,7499,7521);

Select * from EMP where DEPTNO IN(10,20);

Select * from EMP where DEPTNO IN(10,20,30);

PATTERN MATCHING OPERATOR: They are two pattern matching operator.

1. Percentage (%)
2. Under score (_)

Percentage (%): This command is used to select the letters.

Ex: Select * from emp where ename like 'S%'; //starts with letter 'S'

Select * from emp where ename like 'R%'; //ends with letter 'R'
Select * from emp where ename like 'J%S'; // first letter J and last letter S
Select * from emp where ename like '%A%'; //middle letter A
Select * from emp where NOT ename like '%A%'; //not middle letter A

Under Score: This command is also select the letters.

Ex: Select * from emp where ename like '_A%';
Select * from emp where ename like '__A%';

Can we display the rows who's emp name ends with last position 'E'.

Select * from emp where ename like '%E_';
Select * from emp where ename like '____';

Note: Like keyword is used with pattern matching operator.

Select * from emp where deptno NOT IN(10,20);

***DDL (DATA DEFINITION LANGUAGE):**

1. CREATE
2. ALTER
3. DROP
4. TRUNCATE
5. RENAME

ALTER: By using ALTER command we can perform the following task.

1. ADDING A NEW COLUMN
2. DROPPING AN EXISTING COLUMN
3. MODIFYING A COLUMN
4. RENAMING A COLUMN

ADDING A NEW COLUMN:

Syntax: ALTER TABLE <TABLE_NAME> ADD (COL1_NAME DATA TYPE(SIZE),
(COL2_NAME DATA TYPE(SIZE));

Ex: ALTER table student ADD(city varchar2(10));

ALTER table student ADD(state varchar2(10), pincode number(6));

Note: New column can be added only at last.

The new column will have null values.

Update student set city = 'Hyderabad' where sno = 101;

Update student set state = 'Andra pradesh' where sno = 101;

Update student set pincode = 500082 where sno = 101;

DROPING AN EXISTING COLUMN:

Syntax: ALTER TABLE <TABLE_NAME> DROP(COL1_NAME,COL2_NAME);

Ex: ALTER table student drop(state);

ALTER table student drop(city,pincode);

MODIFYING A COLUMN: (increasing/decreasing the size of columns)

Syntax: ALTER TABLE <TABLE_NAME> MODIFY(COL1_NAME DATA TYPE(SIZE));

Ex: ALTER table student modify(Sname varchar2(10));

ALTER table student modify(Sname varchar2(8));

Note: .we can increase and decrease as well as the size of the column.

.We can decrease the column size only when existing column values can fit into new size.

.By using modify keyword we can change the data type of a column.

.Column should be empty to change it's data type.

RENAMEING A COLUMN:

Syntax: ALTER TABLE <TABLE_NAME> RENAME COLUMN<OLD_COL_NAME> TO <NEW_COL_NAME>;

Ex: ALTER table emp rename column SAL TO WAGES;

DROP: This command is used to delete the table from the database.

Syntax: DROP TABLE <TABLE_NAME>;

Ex: DROP table student;

DROP table emp;

TRUNCATE: This command is used to remove all the rows from the table.

Syntax: TRUNCATE TABLE <TABLE_NAME>;

Ex: TRUNCATE table student;

Select * from TAB; // ALL TABLE NAME'S ARE DISPLAYED

*Difference between TRUNCATE and DELETE?

DELETE

- We can Roll Back the data.
- Rows are deleting temporally.
- Where clause can be used.
- Delete is sub language DML.

TRUNCATE

- * We cannot Roll Back the data.
- * Rows are deleting permanently.
- * Where clause cannot be used.
- * Truncate is sub language DDL.

RENAME: This command is used to change the table name.

Syntax: RENAME <OLD_TABLE_NAME> TO <NEW_TABLE_NAME>;

Ex: Rename student To student1;

Rename SALGRADE To GRADE;

NOTE: When we use a Truncate command the table gets dropped and re-created. As the structure is effected is called as a DDL command.

All DDL command are permanent.

All DML command are Temporary.

*To see list of all the tables of a user: Select * from TAB;

***Creating duplicate tables or backup tables:** By using the combination of create and select. We can create copy of a table.

Ex: Create table emp1 AS select * from emp; //total table copy

Create table emp2 AS select * from emp where deptno = 30; // only deptno = 30

Create table emp3 AS select * from emp where 10; // only deptno =10 is copy

Create table emp4 AS select empno, ename, wages, deptno from emp where deptno = 20; //empno,ename,wages,deptno is copied by emp4 table

Create table emp5 AS select *from emp where 1=2; //This mean total table copy

Select * from emp where 1 = 1;

Select * from 'malli' from emp;

***Right click → properties → options and select quick edit modifier and ok.

/ → Run the same query

ED → Open the Buffer command

SET NUM 5

SCOTT is a new user

*****Creating a new user:**

Connect in the database AS DBA.

*username: /AS SYSDBA

*create user: create user malli Identified by malli123; //user and password created

*giving permissions to the user;

*GRANT CONNECT, RESOURCE TO malli; //Grant Succeeded

*SHOW user → To connect the current user.

FUNCTIONS: Functions will manipulate the data items and gives the result. They are two types of functions.

1. Group functions or multiple row functions
2. Scalar functions or single row function

- 1) Group functions or multiple row functions: These functions act on group of rows.
 - i. **AVG**: select AVG(sal) from emp;
 - ii. **SUM**: select SUM(sal) from emp;
 - iii. **MAX**: select MAX(sal) from emp;
 - iv. **MIN**: select MIN(sal) from emp;
 - v. **COUNT(*)**: select COUNT(*) from emp; //Return total no.of rows in the table
 - vi. **COUNT(EXPR)**: Return no.of values present in the column.

Ex: Select COUNT(sal) from emp;

Select COUNT(empno) from emp;

Select COUNT(comm) from emp;

Dual table: It is a dummy table which is generally used to perform some calculation is seeing to the system date and etc. Dual table is collection of one row and one column with 'X' in it.

Ex: Select SYSDATE from dual;

Select 10+20 from dual;

Select 20+40, 50+60 from dual;

- 2) **Scalar functions or single row functions:** Scalar function are decided into

four types. They are given that.

- i. Character functions
- ii. Number functions
- iii. Data functions
- iv. Conversion functions

- i. **Character functions:**

- a. **Upper**: converts into lower case to upper case.

Ex: Select upper('oracle') from dual; //ORACLE

- b. **Lower**: This function is convert to the upper to lower case.

Ex: Select lower('ORACLE') from dual; //oracle

Select ENO, lower('ENAME'), SAL from emp;

- c. **INITCAP**: First letter is capital and reaming letters are small letters.

Ex: Select INITCAP('oracle training') from dual; //Oracle

Select INITCAP('ORACLE TRAINING') from dual; //Oracle

- d. **LENGTH**: Returns length of the string.

Ex: Select LENGTH('oracle') from dual; //length 6

Select LENGTH('MALLIKHARJUNA') from dual; //length 13

Select * from emp where length(ename) = 4;

e. **LPAD**: pads the character towards the left side.

Ex: select LPAD('oracle',10,'z') from dual; //zzzoracle

f. **RPAD**: Rpad the character towards the right side.

Ex: Select RPAD('ORACLE',10,'X') from dual; //ORACLEzzzz

Select LPAD(RPAD('ORACLE',8,'Z',10,'Z') from dual; //Nesting function

g. **LTRIM**:

Ex: Select LTRIM('zzoracle','z') from dual;

h. **RTRIM**:

Ex: Select RTRIM('ZORACLEZZZ','Z') from dual; //ZORACLE

Select RTRIM('oracle') from dual; // oracle

Select RTRIM('ORACLE ') from dual; // ORACLE

Select LENGTH(RTRIM('ORACLE ')) from dual; // length is 6

i. **TRIM**: Removes the specified characters from both sides.

Ex: Select TRIM('z' from 'zzoraclezz') from dual;

Select TRIM(' ORACLE ') from dual;

j. **INSTR**: Returns the position of the string

Ex: Select INSTR('ORACLE','A') from dual; //3

Select INSTR('oracle','H') from dual;

Select INSTR('DATABASE','A') from dual; //2

k. **SUBSTR**: Returns the part of the string.

Ex: Select SUBSTR('ORACLE',2,3) from dual; //RAC

Select SUBSTR('ORACLE',2,4) from dual; //RACLE

Select SUBSTR('ORACLE',2,1) from dual; //R

Select SUBSTR('ORACLE',3,2) from dual; //AC

Select SUBSTR('ORACLE',3,10) from dual; //ACLE

Select SUBSTR('ORACLE',3) from dual; //ACLE

l. **CONCAT**: To join the two words. It will accept only two character.

Ex: Select concat('MAGA','STAR') from dual; //MAGASTAR

Select concat(concat('MAGA','STAR'),'CHIRU') from dual;

Select * from test1;

SNO	SNAME	MARKS
-----	-------	-------

101	ARUN	80
-----	------	----

102	ARUN	80
-----	------	----

103 VIJAY 80

Select * from test1 where SNAME = 'ARUN'; //101 ARUN 80

Select SNO, SNAME, LENGTH(SNAME) from test1;

SNO	SNAME	LENGTH(SNAME)
101	ARUN	4
102	ARUN	5
103	VIJAY	5

Select * from test1 where RTRIM(SNAME) = 'ARUN';

Select * from test1 where TRIM(SNAME) = 'ARUN';

Update test1 set SNAME = TRIM(SNAME);

Select SNO, SNAME, LENGTH(SNAME) from test1;

ii. **Number functions:**

- a. **ABS:** Returns absolute values

Ex: Select ABS(-40) from dual; // 40

Select ABS(40) from dual; //40

- b. **SQRT:** Returns the squarroot values.

Ex: Select SQRT(25) from dual; // 5

Select SQRT(26) from dual; //5.09901951

- c. **MOD(A,B):** Returns the MOD vaues.

Ex: select MOD(10,3) from dual; // 1

- d. **POWER(A,B):**

Ex: Select POWER(2,5) from dual; // 32

- e. **CEIL:**

Ex: Select CEIL(40.9) from dual; //41

Select CEIL(40.2) from dual; //41

Select CEIL(40.5) from dual; //41

- f. **FLOOR:**

Ex: Select FLOOR(40.9) from dual; //40

Select FLOOR(40.2) from dual; //40

- g. **TRUNC:(TRUNCATE)** Remove the decimal points.

Ex: Select TRUNC(40.9) from dual; // 40

Select TRUNC(40.2) from dual; // 40

```
Select TRUNC(40.1234,2) from dual; // 40.12  
Select TRUNC(685.195364,3) from dual; // 685.195  
Select TRUNC(6854,-1) from dual; // 6850  
Select TRUNC(6854,-2) from dual; // 6800  
Select TRUNC(7777,-3) from dual; // 7000
```

h. **ROUND**: Rounds of the nearest value.

Ex: Select ROUND(40.9) from dual; //41
Select ROUND(40.2) from dual; //40
Select ROUND(123.863,2) from dual; //123.86
Select ROUND(123.868,2) from dual; //123.87
Select ROUND(856.81766,3) from dual; //856.818
Select ROUND(123,-1) from dual; //120
Select ROUND(140,-2) from dual; //100
Select ROUND(127,-3) from dual; //130
Select ROUND(17763,-3) from dual; //18000

i. **GREATEST**:

Ex: Select GREATEST(100,200,300) from dual; // 300

j. **LEAST**:

Ex: Select LEAST(100,200,300) from dual; // 100

iii. **Datafunctions**: They are four data functions.

- A. ADD_MONTHS
- B. MONTHS_BETWEEN
- C. NEXT_DAY
- D. LAST_DAY

A.**ADD_MONTHS**: ADD_MONTHS of months to the given date.

Ex: Select ADD_MONTHS(SYSDATE,12) from dual; // 16-JUN-13
Select ADD_MONTHS('11-APR-05',3) from dual; // 11-APR-05

B.**MONTH_BETWEEN**: Returns number of months b/w given the two months.

Ex: Select MONTHS_BETWEEN('11-JAN-05','11-JAN-04') from dual; // 12
Select MONTHS_BETWEEN('11-JAN-04','11-JAN-05') from dual; // -12
Select EMPNO, ENAME, SAL, HIREDATE from emp; //display emp table
Select EMPNO, ENAME, SAL, MONTHS_BETWEEN(SYSDATE,HIREDATE) from emp;
Select EMPNO, ENAME, SAL, MONTHS_BETWEEN(SYSDATE,HIREDATE)/12 from emp;
Select EMPNO, ENAME, SAL, ROUND(MONTHS_BETWEEN(SYSDATE,

HIREDATE)/12) from emp;

Select EMPNO, ENAME, SAL, ROUND(MONTHS_BETWEEN(SYSDATE, HIREDATE)/12) EXP from emp;

C. **NEXT_DAY**: Returns date of the specified date.

Ex: Select NEXT_DAY(SYSDATE,'MONDAY') from dual; // 18-JUN-12

Select NEXT_DAY('11-JAN-05','MONDAY') from dual; // 17-JAN-05

D. **LAST_DAY**: Returns the last day of the month.

Ex: Select LAST_DAY(SYSDATE) from dual; // 30-JUN-12

Select LAST_DAY('11-FEB-05') from dual; // 28-FEB-05

iv. **Conversion functions:**

Conversion functions are one data type to another data type conversion.

They are three conversion functions.

- TO_CHAR
- TO_NUMBER
- TO_DATE

1. **TO_CHAR**: This function is having two functionalities.

a. **Number to_char**: This function is used only \$ or u-rows and number is used 9.

Ex: Select eno,ename, TO_CHAR(sal,'9,999') from emp;

Select eno,ename, TO_CHAR(sal,'99,99') from emp;

Select eno,ename, TO_CHAR(sal,'\$9,999') from emp;

Select eno,ename, TO_CHAR(sal,'8,888') from emp; // invalid number format

b. **Date to_char**:

Ex: Select eno,ename,hiredate from emp;

Select eno,ename, TO_CHAR(HIREDATE,'DD-MM-YY') from emp;

Select eno,ename, TO_CHAR(HIREDATE,'DD-MM-YYYY') from emp;

Select SYSDATE from dual; // 18-JUN-12

Select TO_CHAR(SYSDATE,'DD-MONTH-YY') from dual; // 18-JUN-12

Select TO_CHAR(SYSDATE,'DAY') from dual; // Monday

Select TO_CHAR(SYSDATE,'YYYY') from dual; // 2012

Select TO_CHAR(SYSDATE,'MM') from dual; // 06

Select TO_CHAR(SYSDATE,'DDD') from dual; // 170

Select TO_CHAR(SYSDATE,'DD') from dual; // 18

Select TO_CHAR(SYSDATE,'MON') from dual; // MONDAY

Select TO_CHAR(SYSDATE,'DY') from dual; // mon

Select TO_CHAR(SYSDATE,'DD-MM-YY HH:MI:SS') from dual; //18-06-12 12:40:44

Select * from emp where TO_CHAR(HIREDATE,'YYYY') = '1981';

```
Select * from emp where TO_CHAR(HIREDATE,'YYYY') = '1980';
Select * from emp where TO_CHAR(HIREDATE,'MON') = 'DEC';
```

2.TO_NUMBER:

Ex: Select TO_NUMBER(LTRIM('\$1400','\$')) + 10 from dual; // 1410

3.TO_DATE:

This function is used to convert character values to data value.

Ex: ADD_MONTHS

```
Select ADD_MONTH('11-JAN-05',2) from dual;
```

```
Select ADD_MONTH('11-JANUARY-2005 11:45 A.M.','DD-MONTH-YYYY
HH:MI A.M'),2) from dual;
```

```
Select ADD_MOONTHS(TO_DATE('11-01-2005 11:45 A.M.',
'DD-MM-YYYY HH:MI A.M.'),2) from dual; //11-MAR-2005
```

*implicit conversion:

Ex: Select * from emp where DEPTNO = '10';

```
Select sum(sal) from emp where deptno = '10'; //8750
```

```
Select sum(sal) from emp where deptno = '20'; //10875
```

```
Select sum(sal) from emp where deptno = '30'; // 9400
```

***Group By clause:** Group By clause is used to divided rows into several group. So that we can apply group function on each group.

Ex: Select deptno, sum(sal) from emp Group By deptno;

<u>Deptno</u>	<u>Sal</u>
30	9400
20	10875
10	8750

Select deptno,min(sal),max(sal) from emp Group By deptno;

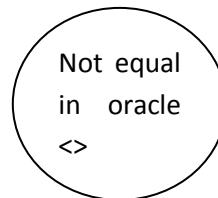
<u>Deptno</u>	<u>Min</u>	<u>Max</u>
30	950	2850
20	800	3000
10	1300	5000

Select job, avg(sal) from emp Group By job;

Select job,count(*) from emp Group By job;

Select job,count(job) from emp Group By job;

Select Deptno, sum(Sal), min(Sal), max(Sal), avg(Sal), count(*) from emp
Group By deptno;



We can use the combination of where clause and Group By clause.

First where clause is executed on the result of where clause Group By clause is apply.

Select deptno, sum(sal) from emp where deptno <> 10 Group By deptno;

Select deptno, job, sum(sal) from emp Group By deptno, job;

***Rule of group by clause:** All the column in the select of list should use group functions or should be included in group by clause.

Select deptno, sum(sal), ename from emp Group By deptno;

Error: Not a Group By Expression

***Having clause:** (to use Group By clause)

Having clause is used to filter the output from Group By clause.

Ex: Select deptno, sum(sal) from emp Group By deptno having sum(sal) > 9000;

<u>Deptno</u>	<u>Sum(sal)</u>
30	9400
20	10875

***Order By clause:**

Order By clause is used to arrange the rows in the table.

By default order by clause ascending order.

Null values are arranged in the last.

Ex: Select * from emp Order By sal;

Select * from emp Order By ename;

Select * from emp Order By HIREDATE; //Chronological order 1980....1985

Select * from emp Order By eno;

Select * from emp Order By job,sal;

Select * from emp Order By sal DESC; //descending order by depending the query

Note: Order by clause should be the last change of the query.

Select deptno, sum(sal) from emp Group By deptno Having sum(sal) > 9000

Order By sum(sal) DESC;

Select deptno, sum(sal) from emp where ename <> 'King' Group By deptno;

Select deptno, sum(sal) from emp where ename <> 'King' Group By deptno

Having sum(sal) > 9000;

Select deptno, sum(sal) from emp where ename <> 'King' Group By deptno

Having sum(sal) > 9000 Order By sum(sal) DESC;

*****Interview questions*****

1. What is SQL?

SQL transfer Structured Query Language are also called as SEQUEL.

2. List out sub language of SQL?

They are 5 sub languages DDL,DML,DRL/DQL,TCL,DCL .

3. What is different between char and varchar2?

Char is a fixed size and varchar2 is a not a fixed size.

4. What is projection?

Selecting specific columns is projection.

5. How can we filter the rows in the table by use the Where, Group BY, Having, Order By clause?

Select deptno, sum(sal) from emp where ename <> 'KING' Group By deptno
Having sum(sal) > 9000 Order By sum(sal) DESC;

6. What is column Alias?

Providing the duplicate to column Name. This is not a permanent.

7. Can we perform a arithmetic operation by using dual?

Select 10 + 20 Result from dual;

8. What is dual table?

Dual table is dummy table to calculate the some problems. This is one column and one row. Specified to 'X'

9. Write a query to display current date along with HH:MI:SS?

Select To_Char(sysdate,'DD-MON-YYYY HH:MI:SS') from dual?

10. Write a query to see the current date?

Select sysdate from dual;

11. Which operator is used to accept the values from the user?

INSERT, UPDATE, CREATE and etc.

12. How can you see all the table which are in the data base?

Select * from TAB

13. Which command is used to remove the table from the data base?

Drop command is used to the table.

14. What is different between delete and truncate command?

Delete to the table and getting the roll back. Truncate is used not possible the table roll back.

15. Which operator is used to retrieve the rows based on null values?

IS NULL

16. In how many ways we can rename all the rows from the table?

They are two ways Delete and Truncate

17. How can we create copy of a table?

Create table emp1 AS select * from emp;

Create table emp2 AS select * from emp where deptno = 30;

18. Write a query to display no.of rows in the table?

BY using count(*)

Select count(*) from emp;

19. What is different between count(*) and count(Expr)?

*is total table. Expr is only one column to count the table.

20. What is difference between group functions and scalar function?

Group functions will act on total table and scalar functions will act on one row.

21. What is a use of Group By clause?

Group By clause will decided into several groups.

22. How can we filter the rows of Group By clause?

Having clause is used to filter the data.

23. Which clause is used to arrange the rows in the table?

Order By clause is used to arrange.

24. Which clause should be the last clause of the query?

Is order By caluse.

***Any operation performed on null will result to null values.

25. What is a TOAD?

Tool for Oracle Application Development.

***INTEGRITY CONSTRAINTS:**

Constraints are rules which are applied on tables.

Constraints helps in improving the accuracy and quality of the data base.

They are five types of constraints.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY or REFERENTIAL INTEGRITY CONSTRAINTS
5. CHECK

Constraints can be created at two levels

- a. Column level
- b. Table level

1. **NOT NULL:** This constraints will not accept null values.

NOT NULL constraints can be created only at column level.

Ex: Create table student1(Sno number(3) NOT NULL, Sname varchar2(10),

Marks number(3));

Insert into student1 values(101,'Arun',50);
Insert into student1 values(101,'Arun',NULL);
Insert into student1 values(101,NULL,50);
Insert into student1 values(NULL,'Arun',50); \times

Error: cannot insert into null to scott, student1, sno.

*Create table student2(Sno number(3) NOT NULL, Sname varchar2(10),Marks number(3) NOT NULL);

Insert into student2 values(101,'Arun',50);
Insert into student2 values(101,'Arun',NULL); \times
Insert into student2 values(101,NULL,50);
Insert into student2 values(NULL,'Arun',50); \times

2. UNIQUE CONSTRAINTS: This constrains will not accept duplicate values.

This constrains can be created at column level as well as table level.

Ex: Creating unique constraint at column level.

* Create table student3(Sno number(3) UNIQUE, Sname varchar2(10), Marks number(3));

Insert into student3 values(101,'Arun',50);
Insert into student3 values(102,'Arun',NULL);
Insert into student3 values(101,NULL,50); \times
Insert into student3 values(NULL,'Arun',50);
Insert into student3 values(NULL,'Arun',50);

Note: UNIQUE constraint will accept multiple will values.

*Create table student4(Sno number(3) UNIQUE, Sname varchar2(10),Marks number(3) UNIQUE);

Insert into student4 values(101,'Arun',50);
Insert into student4 values(102,'Arun',50); \times

Creating unique constraint at table level:

Ex:

* Create table student5(Sno number(3), Sname varchar2(10),Marks number(3),UNIQUE(Sno));

Insert into student5 values(101,'Arun',50);
Insert into student5 values(102,'Arun',NULL);
Insert into student5 values(101,NULL,50); \times
Insert into student5 values(NULL,'Arun',50);
Insert into student5 values(NULL,'Arun',50);

Note: There is no difference when a constraint at column level or table level.

****3. PRIMARY KEY CONSTRAINTS:**

A primary key constrains of a combination of NOT NULL and UNIQUE.

A primary key constraints will not accept null values as well as duplicate values.

Primary key column is used to uniquely identify every row in a table.

A table can have only one primary key.

Primary key constraints can be created at column level or table level.

Create primary key constraint at column level:

Ex: Create table student6(Sno number(3) PRIMARY KEY, Sname varchar2(10), Marks number(3));

Insert into student6 values(101,'Arun',50);

Insert into student6 values(102,'Arun',50);

Insert into student6 values(101,Arun,50); \times

Insert into student6 values(NULL,'Arun',50); \times

Insert into student6 values(103,'Arun',50);

Create Primary key constraint at table level:

Ex: Create table student7(Sno number(3),Sname varchar2(10),Marks number(3),PRIMARY KEY(Sno));

****3.1.COMPOSITE PRIMARY KEY:**

When primary key is applied to multiple columns it is called composite primary key.

Composite primary key can be applied only at table level.

***Creating composite primary key constraint at table level:**

Ex: Create table student9(Surname varchar2(10),Firstname varchar2(10),Marks number(3),
PRIMARY KEY(Surname,Firstname));

Insert into student9 values('xyz','Arun',40);

Insert into student9 values('xyz','Kiran',40);

Insert into student9 values('mno','Arun',40);

Insert into student9 values('xyz','Kiran',40); \times

Insert into student9 values(NULL,'Arun',40); \times

Insert into student9 values('abc',NULL,40); \times

*****4.FOREIGN KEY CONSTRAINTS or REFERENTIAL INTEGRITY:**

These constraints establish relationship between tables.

This relationship is called as parent and child relationship. It is also called master detail relationship.

A foreign key column in the child table will only accept values which are their the

primary key column or unique column of parent table.

Creating the parent table:

```
Create table school(sno number(3),Sname varchar2(10),Marks number(3),
primary key(sno));
```

Insert Rows parent table:

```
Insert into school values(101,'Arun',90);
Insert into school values(102,'Kiran',92);
Insert into school values(103,'Amit',45);
```

Creating the child table:

```
Create table library(sno number(3) REFERENCES school(sno),Book_name varchar2(10));
```

Insert Rows child table:

```
Insert into library values(102,'java');
Insert into library values(103,'c++');
Insert into library values(103,'oracle');
Insert into library values(108,'dotnet'); //error
Insert into library values(Null,'DBA'); //valid
```

Foreign key column name need not match with primary key column name or unique column name. But the data type should match.

To establish relationship it is mandatory that the parent table should have primary key constraint or at least unique constraints.

Delete from school where sno = 101;

1 row deleted

Delete from school where sno = 102; //error

Message: child record found

Note: we can not delete to the row from the parent table in the corresponding value existing child table.

***Using on delete cascade:**

When using on delete cascade. We can delete the rows from the parent table and the corresponding child table rows deleted automatically.

Create the parent table:

```
Create table school1(sno number(3),Sname varchar2(10),Marks number(3),
primary key(sno));
```

Insert Row parent table:

```
Insert into school1 values(101,'Arun',90);
```

```
Insert into school1 values(102,'Kiran',92);
Insert into school1 values(103,'Amit',45);
```

Creating the child table:

```
Create table library1(sno number(3) REFERENCES school1(sno) on delete cascade,
Book_name varchar2(10));
```

Insert Rows child table:

```
Insert into library1 values(102,'java');
Insert into library1 values(103,'c++');
Insert into library1 values(103,'oracle');
Insert into library1 values(108,'dotnet'); //error
Insert into library1 values(Null,'DBA'); //valid
```

Delete from student1 where sno = 101; //valid

1 row deleted

Delete from student1 where sno = 102; //valid

1 row deleted

One row will be deleting from parent table.

One row will be deleting from child table automatically.

***On delete set null:** When delete the row from parent table. The corresponding value will be changed to null.

Create the parent table:

```
Create table school2(sno number(3),Sname varchar2(10),Marks number(3),
primary key(sno));
```

Insert Row parent table:

```
Insert into school2 values(101,'Arun',90);
Insert into school2 values(102,'Kiran',92);
Insert into school2 values(103,'Amit',45);
```

Creating the child table:

```
Create table library2(sno number(3) REFERENCES school2(sno) on delete set null,
Book_name varchar2(10));
```

Insert Rows child table:

```
Insert into library2 values(102,'java');
Insert into library2 values(103,'c++');
Insert into library2 values(103,'oracle');
Insert into library2 values(108,'dotnet'); //error
```

Insert into library2 values(Null,'DBA'); //valid
Delete from school2 where sno = 102; //valid
One row from parent table is deleted.
Foreign key column value 102 is changed to null.

***Create foreign key constraint at table level:**

Create table school3(sno number(3),Sname varchar2(10),Marks number(3),
primary key(sno));

Insert Row parent table:

Insert into school3 values(101,'Arun',90);
Insert into school3 values(102,'Kiran',92);
Insert into school3 values(103,'Amit',45);

Creating the child table:

Create table library3(Rollno number(3),Book_name varchar2(10),Foreign key (Rollno)
REFERENCES school3(sno) on delete cascade);

***Check constraint:** Check constraint is used to define domain of a column. Domain of column means values a column can store.

Create table student4(sno number(3),Sname varchar2(10),Marks number(3));
Insert into student4 values(101,'ARUN',66);
Insert into student4 values(102,'ARUN',80);
Insert into student4 values(103,'ARUN',166);

***Domain:** Create table student5(sno number(3),Sname varchar2(10),
Marks number(3) check(marks <=100));

Insert into student5 values(101,'ARUN',60);
Insert into student5 values(102,'ARUN',80);
Insert into student5 values(103,'ARUN',160); //Error
Msg: check constraint violated
Insert into student5 values(101,'ARUN',-160);

Create table student6(sno number(3), Sname varchar2(10),
Marks number(3) check(marks between 0 AND 100));
Insert into student6 values(101,'ARUN',60);
Insert into student6 values(101,'ARUN',80);
Insert into student6 values(101,'ARUN',160); //Error

```
Create table student7(sno number(3), Sname varchar2(10);
                      Marks number(3),
                      City varchar2(10) check(city IN('HYDERABAD','CHENNAI','DELHI'));
```

```
Insert into student7 values(101,'ARUN',66,'HYDERABAD');
Insert into student7 values(101,'ARUN',66,'DELHI');
Insert into student7 values(101,'ARUN',66,'CHENNAI');
Insert into student7 values(101,'ARUN',66,'NELLORE'); //Error
```

```
Create table student8(sno number(3),Sname varchar2(10) check(Sname = upper(sname)),
                      Marks number(3));
```

```
Insert into student8 values(101,'ARUN',60);// Valid
Insert into student8 values(101,'ARuN',60);// error
Insert into student8 values(101,'arun',60); // error
```

*Check constraint at table level:

```
Create table student8(sno number(3),Sname varchar2(10),Marks number(3),
                      Check (sname = upper(Sname)));
```

Note: Every constraint will have a constraint name in the format of
SYS_Cn(where N is number).

Ex: SYS_c004525, SYS_c004526

*Data Dictionary tables:

Select TABLE_NAME, CONSTRAINT_NAME,CONSTRAINT_TYPE from USER_CONSTRAINT;
Set of predefined table which constraints meta data information are called as data dictionary table.

Ex: USER_CONSTRAINT

Query to find constraint_name and constraint_type of the table.

```
Select TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE from USER_CONSTRAINT
where TABLE_NAME = 'student5'; or 'student7';
```

We can also provide user defined constraint_name

Ex: Create table student9(sno number(3) CONSTRAINT MY_UN UNIQUE,
 Sname varchar2(10), Marks number(10));

In the above query “MY_UN” is the constraint_name.

*ALTER:

*Adding Constraints: Alter command is used to add a constraint to an Existing table.

Ex: Create table Student10(sno number(3),Sname varchar2(10),Marks number(3));

Insert into student10 values(101,'Arun',60);

Insert into student10 values(102,'Arun',80);

Insert into student10 values(103,'Arun',90);

ALTER table student10 ADD(Primary key(Sno));

*Dropping a constraint:

Alter command is used to drop a constraint to an existing table.

Ex: Alter table student10 DROP Primary key;

*Unique Constraint:

Ex: Create table Student11(sno number(3), Sname varchar2(10),Marks number(3));

Insert into student11 values(101,'Arun',60);

Insert into student11 values(102,'Arun',80);

Insert into student11 values(103,'Arun',90);

ALTER table student11 ADD(Unique(sno));

ALTER table student11 DROP Unique(Sno);

***JOINS: joins are used to retrieve the data from multiple tables.

Types of Joins:

1. EQUI_JOIN
2. NON EQUI_JOIN
3. SELF JOIN
4. OUTER JOIN
 - 4.1 Right outer join
 - 4.2 Left outer join
 - 4.3 Full outer join

1.EQUI JOIN: when tables are joined basing on a common column it is called EQUI_JOIN.

Ex: select empno, ename, dname from emp, dept where emp.deptno = dept.deptno;

output: **EMPNO** **ENAME** **DNAME**

7369	SMITH	RESEARCH
7499	ALLEN	SALES
7521	WARD	SALES

Note: We need to mention join conditions in the where clause.

In EQUI_JOINS we along use to equal to operator in join condition.

Ex: Selete empno, ename, sal, job, dname, loc from emp, dept where

emp.deptno = dept.deptno;

Select empno, ename, sal, deptno, dname, loc from emp, dept where
emp.deptno = dept.deptno; // error

Select empno, ename, sal, emp.deptno, dname, loc from emp, dept
where emp.deptno = dept.deptno; //valid

Note: we need to mention table name dot column(emp.deptno) name for the common column to resolve the any table.

The common column can be retrieved from any of the table.

We can filter the data from the result of join.

Ex: Select empno, ename, sal, emp.deptno, dname, loc from emp, dept
where emp.deptno = dept.deptno AND sal > 2000;

To improve the performance of the join we need mention table name dot column name for all the columns.

Ex: Select emp.empno, emp.ename, emp.sal, emp.deptno, dept.dname, dept.loc
from emp,dept where emp.deptno = dept.deptno AND sal > 2000;

*Table alias:

Table alias is an alternate name given to a table.

By using a table alias length of the table reduces and at the same time performance is maintains.

Table alias are create in same clause can be used in select clause as well as where clause.

Table alias is temporary once the query is executed the table alias are losed.

Ex: Select E.Empno, E.Ename, E.sal, E.deptno, D.Dname, D.loc from emp E, Dept D
where E.deptno = D.deptno;

*Join the multiple tables(3 tables):

Select * from Areas;

City State

Newyork AP

Dallas Mh

Ex: Select E.empno, E.ename, E.sal, D.dname, A.state from emp E, dept D, Areas A
where E.deptno = D.deptno AND D.loc = A.city;

Note: To join 'n' tables we need n-1 conditions.

***NON EQUI JOIN:** When we do not use NON EQUI JOIN to operator in the join condition is NON EQUI JOIN.

Ex: Select * from SALGRADE;

GRADE LOSAL HISAL

1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

Select e.empno, e.ename, e.sal, s.grade from emp e, salgrade s
where e.sal BETWEEN s.losal AND hisal;

<u>EMPNO</u>	<u>ENAME</u>	<u>GRADE</u>
7369	SMITH	1
7876	ADAMS	1
7900	JAMES	2

Select e.empno, e.ename, s.grade from emp e, salgrade s
where e.sal BETWEEN s.losal AND s.hisal AND s.grade = 4;

***SELF JOIN**: When a table is joining to it self it is called self join. In self joins we need to create two table aliases for the same table.

Select empno, ename, job, mgr, from emp;

Select e.empno, e.ename, e.job, m.ename from emp e, emp m where e.mgr = m.empno;

<u>Empno</u>	<u>Ename</u>	<u>Job</u>	<u>Ename</u>
7902	FORD	ANALYST	JONES
7869	SCOTT	CLERK	JONES
7900	JAMES	SALESMAN	BLAKE

***CARTESIAN PRODUCT:**

When tables are joined without any join condition it is called Cartesian product.
In the result we get all possible combination.

Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc
from emp e, dept d; //14*4=56 rows are selected

***ANSI JOINS**: They are the three types.

1. **Inner joins**: It is same as Equi join.

Ex: Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc
from emp e INNER JOIN dept d ON(e.deptno = d.deptno);

2. **Natural join**: It is same as Equi join.

Ex: Select empno, ename, sal, deptno, dname, loc from NATURAL JOIN dept;

3. CROSS PRODUCT/CROSS JOIN: It is same as Cartesian product.

Ex: Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc
from emp e CROSS JOIN dept d; //14*4 = 56 rows are displayed.

***DEFAULT:**

Ex: Create table stu1(sno number(3),Sname varchar2(10),Marks number(3)
default 100,Doj Date DEFAULT sysdate);

```
Insert into stu1(sno, sname) values(101,'malli');  
Insert into stu1 values(102,'ARUN',40,'11-JAN-09');  
Insert into stu1 values (103,'KIRAN',NULL,'12-FEB-10');
```

SNO	SNAME	MARKS	DOJ
101	malli	100	26-JUN-12
102	ARUN	40	11-JAN-09
103	KIRAN		12-FEB-10

***SUPER KEY:** Combination of columns which can be used unique key identify every row is called as super key.

Table object

Column Attributes

Row Tuple/Record

***OUTER JOINS:** It is extension of EQUI JOINS.

In outer joins we get match as well as non matching rows.

(+) This called as outer join operator.

1. RIGHT OUTER JOIN:

SQL Syntax:

```
Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc from emp e, dept d  
where e.deptno(+) = d.deptno; //14 + 1 = 15 rows
```

empno	ename	sal	deptno	dname	loc
7900	james	950	30	sales	chicago
8963	adams	1400	20	clerk	newyork
6798	adams	2000	10	sales	india

***ANSI SYNTAX OF RIGHT OUTER JOIN:**

ANSI SYNTAX:

```
Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc  
from emp e RIGHT OUTER JOIN dept d ON(e.deptno = d.deptno);
```

2. LEFT OUTER JOIN:

SQL Syntax:

Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc from emp e, dept d
where e.deptno = d.deptno(+); //14+3 = 17 row displayed

ANSI SYNTAX OF LEFT OUTER JOIN:

ANSI SYNTAX:

Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc
from emp e LEFT OUTER JOIN dept d ON(e.deptno = d.deptno);

3. FULL OUTER JOIN:

ANSI SYNTAX:

Select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc
from emp e FULL OUTER JOIN dept d ON(e.deptno = d.deptno);
//14 + 2 + 3 = 19 rows are displayed.

*SET OPERATORS: Set operators are used to retrieve the data from multiple tables.

They are different types.

1. UNION:

Select * from student10;

<u>Sno</u>	<u>sname</u>	<u>marks</u>
101	Arun	40
102	Arun	50
103	Arun	69

Select * from student20;

<u>Sno</u>	<u>sname</u>	<u>marks</u>
103	Arun	90
104	Arun	60

Union Syntax: (no duplicates)

Select sno from student 10

Union //0/p sno: 101 102 103 104

Select sno from student 20;

2. UNION ALL:

Union All Syntax: (All rows)

Select sno from student 10

Union All // o/p sno: 101 102 103 103 104

Select sno from student 20;

3. INSERT SECT:

Insert Sect Syntax: (common rows)

Select sno from student 10

Insert Sect // o/p sno: 103

Select sno from student 20;

4. MINUS:

Select sno from student 10

Minus //o/p sno: 101,102

Select sno from student 20;

Select sno from student 20

Minus // o/p sno: 104

Select sno from student10;

RULES OF SET OPERATORS:

1. Number of columns used in the query should match.
2. Column data type should match for the queries in set operators.

Select empno from emp

Union

Select sno from student10

Union

Select deptno from dept; // valid

INTERVIEWS QUESTIONS

1. What is need for Integrity constraint?

Constrains are rules which are applied on tables.

2. List out types of constraints?

They are 5 types NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, and CHECK.

3. In how many level constraints can be created?

Those are two levels i.e. column level and table level.

4. Which constraint can be created?

The constraint created not null.

5. Does not null constraints accept duplicate values?

Yes

6. Which constraint is used to unique for every row in the table?

Primary key

7. What is composite primary key?

When primary key is applied on multiple columns it is called composite primary key. Composite primary key can be applied only at table level.

8. Can a table name two primary key?

It is not possible.

9. What is foreign key constraint explain?

This foreign key is established in parent table and child table relationship.

10. Can we establish a parent & child relationship without having constraint in the parent table?

On

11. Con you explain change related to foreign key on delete cascade on delete set null constraint?

Foreign key column in the child table will only accept values which are their the primary key column or unique column.

We can delete the rows from the parent table and the corresponding child table rows deleted automatically.

When we delete row from parent table. The corresponding values will be changed to null.

12. Does every constraint it have constraint name?

13. How can you know constraint name and combination type apply for a table?

By using user constraint.

14. Is there any difference when a constraint is created at column level or table level?

No difference.

15. Can you provide user defined constraint name?

Yes

16. What are data dictionary tables?

Predefined tables or user constraints

17. What is the need for join?

To retrieve the multiple tables

18. What is EQUI join?

When tables are joined basing on a common column it is called EQUI_JOIN.

19. How many conditions are required to join 'n' tables?

We need to n-1 conditions.

20. How can be display matching as well as non matching rows?

By using outer joins.

21. What is outer join operator?

(+)

22. What is Cartesian product?

All possible in the table matching

23. What is difference between union and union all?

The union set operator display only original values and union all set operator is display all values. Duplicate values also.

****TCL (Transaction Control Language)**: It is collection of three commands. They are

1. COMMIT
2. ROLLBACK
3. SAVE POINT

***Commit**: This command is used to make changes permanent to the data base.

Syntax: COMMIT;

Ex:

```
Create table student(sno number(3),  
                    Name varchar2(10),  
                    Marks number(3));
```

```
Insert into student values(300,'Arun',69);
```

```
Insert into student values(301,'Kiran',69);
```

```
Insert into student values(302,'Naga',69);
```

```
Select * from student300;
```

```
Create table student1(sno number(3),Name varchar2(10), Marks number(3));
```

```
Insert into student1 values(300,'Arun',69);
```

```
Insert into student1 values(301,'Kiran',69);
```

```
Insert into student1 values(302,'Naga',69);  
COMMIT;
```

In three ways we can make changes permanent to the data base.

1. By using command COMMIT
2. By using DDL command
3. By using the environment using EXIT command

***ROLLBACK:** The rollback will undo the changes which are not permanent.

Syntax:

```
ROLLBACK;
```

Ex:

```
Create table student2(sno number(3),  
                      Name varchar2(10),  
                      Marks number(3));
```

```
Insert into student2 values(300,'Arun',69);
```

```
Insert into student2 values(301,'Kiran',69);
```

```
Insert into student2 values(302,'Naga',69);
```

```
COMMIT;
```

```
Insert into student2 values(304,'Arun',69);
```

```
Insert into student2 values(305,'Kiran',69);
```

```
Select * from student2; //display 5 rows
```

```
ROLLBACK;
```

```
Select * from student2; //display 3 rows there are permanently
```

***SAVE POINT:** Save points are logical marking given for series of transaction.

Instead of complete rollback we can rollback to a save point.

Syntax: SAVEPOINT<SAVEPOINT_NAME>;

Ex: Create table student3(sno number(3), Name varchar2(10), Marks number(3));
Insert into student3 values(300,'Arun',69);
Insert into student3 values(301,'Kiran',69);
Insert into student3 values(302,'Naga',69);
Insert into student3 values(303,'Arun',69);

```
Insert into student3 values(304,'Kiran',69);  
SAVEPOINT A;  
Insert into student3 values(305,'Naga',69);  
Insert into student3 values(306,'Kiran',69);  
Insert into student3 values(307,'Naga',69);
```

```
Select * from student3; //8 rows displayed
```

```
ROLLBACK;
```

```
Select * from student3; //5 rows are displayed
```

```
Create table student4(sno number(3), Name varchar2(10), Marks number(3));
```

```
Insert into student4 values(300,'Arun',69);  
Insert into student4 values(301,'Kiran',69);
```

```
SAVEPOINT P;
```

```
Insert into student4 values(302,'Naga',69);  
Insert into student4 values(303,'Naga',69);
```

```
SAVEPOINT Q;
```

```
Insert into student4 values(304,'Naga',69);  
Insert into student4 values(305,'Naga',69);
```

```
Select * from student4; // 6 rows
```

```
ROLLBACK;
```

```
Select * from student4; //0 rows
```

Note: All the save points are lost when the DB is permanent.

*****DCL (Data Control Language):**

They are two Data Control Languages.

1. GRANT
2. REVOKE

Schema: Schema is a memory location which is associated to a user.

It is collection of objects.

Privilege: privileges are permissions (rights given to a user)

They are two types of privileges.

1. System Privileges
2. Object Privileges

***System Privileges:** These privileges are given by DBA to user.

***Object Privileges:** These Privileges are given by one user to another user.

***GRANT:** Grant command is used to Grant the privileges to the user.

Syntax:

```
GRANT <PRIVILEGE_NAME1>,<PRIVILEGE_NAME2> TO <USER_NAME>;
```

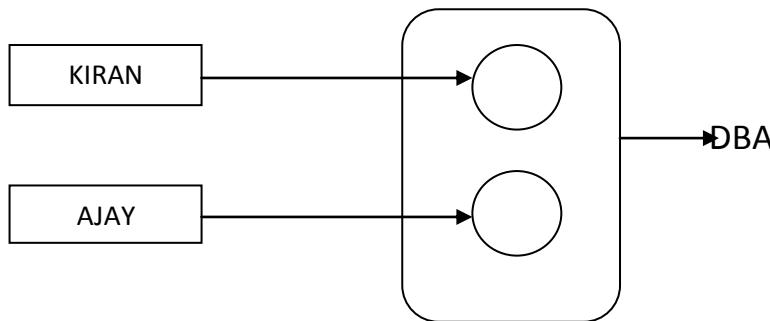
Ex:

```
Create user kiran IDENTIFIED by kiran123;
```

```
Create user naga IDENTIFIED by naga123;
```

```
DBA> GRANT CONNECT, RESOURCE to kiran;
```

***Object Privileges:** These privileges are given by one user to another user.



```
KIRAN> create table student(Sno number(3),
                           Name varchar2(10),
                           Marks number(3));
```

```
KIRAN> insert into student values(101,'Arun',99);
```

```
KIRAN> insert into student values(101,'Anil',97);
```

```
KIRAN> insert into student values(101,'Anitha',95);
```

```
COMMIT;
```

```
Select * from student;
```

```
AJAY> select * from student; //There is no response
```

```
KIRAN> GRANT select ON student TO AJAY; // KIRAN given privileges to AJAY
```

Examples of object privileges are select, insert, update, drop and alter etc.

```
KIRAN> GRANT insert, update ON student TO AJAY;  
AJAY> insert into KIRAN.student values(104,'Nandini',89);  
AJAY> update KIRAN.student set sno = 102 where ename = 'Arun';  
KIRAN> GRANT ALL ON student TO AJAY,ANIL;  
KIRAN> GRANT select ON student TO public;
```

****REVOKE:** This command is used to get back the privileges which are granted.

Syntax: REVOKE<privilege_name><privilege_name> ON <table_name> from <user_name>;

Ex:

```
KIRAN> REVOKE select, insert ON student from AJAY;  
KIRAN> REVOKE ALL ON student from AJAY_ANIL;  
KIRAN> REVOKE select ON student from public;
```

****DML** (data manipulation language)

MERGE: MERGE command is used as a combination of insert and update.

Select * from student10; // 3 rows are displayed

SNO	SNAME	MARKS
101	Arun	30
102	Anil	40
103	Kiran	50

Select * from student20; // 2 rows are selected

SNO	SNAME	MARKS
101	James	90
105	Smith	50

```
KIRAN> merge into student10 s1  
2>      using student20 s2  
3>      on(s1.sno = s2.sno)  
4>      when matched  
5>      then updated set sname = s2.sname, marks = s2.marks  
6>      when not matched  
7>      then insert(sno, sname, marks) values(s2.sno, s2.sname, s2.marks);
```

o/p: 2 rows merge

SNO	SNAME	MARKS
101	James	90

102	Anil	40
103	Kiran	50
105	Smith	50

Note: There will be no changes student20 table.

Delete from emp where ename = null;

Select * from emp where eno = 7369 or eno = 7499 or eno = 7521;

Select * from emp where eno IN(7369,7499,7521);

****PSEUDO COLUMNS:**

1. **ROWNUM:** It is ROWNUM is a pseudo column which starts with one and increment by 1. (1 and 1+1)

Ex:

Select Rownum, empno, ename, sal, deptno from emp;

Rownum values are temporary.

Rownum values are generated when query is executed.

Rownum values generation always starts with one and increment by one.

*Query to display first three rows from emp table?

Select * from emp where Rownum <=3;

Select * from emp where Rownum <=10;

*write a query to display 5th row of emp table?

Select * from emp where Rownum <=5

Minus

Select * from emp where Rownum <=4; //5th row is display

*write a query to display 3rd row to 7th row?

Select * from emp where Rownum <=7

Minus

Select * from emp where Rownum <=2; //3rd to 7th row display

****ROWID:**

ROWID is pseudo column which contains hexadecimal values.

ROWID value indicates the address where the row is stored in the database.

ROWID values are permanent.

Ex:

Select ROWID, empno, ename, sal, deptno from emp;

*Difference between ROWNUM and ROWID?

<u>ROWNUM</u>	<u>ROWID</u>
1.Rownum values starts with 1 and increment by one.	1. Rowid's are hexadecimal values.
2.Rownum values are temporary.	2. Rowid values are permanent.
3.Rownum values are generated when query is exiecuted.	3. The Rowid values are generated when Row is created or inserted.

Create table student(Sno number(3),

Sname varchar2(10),
Marks number(3));

Insert into student values(101,'Arun'60);

Insert into student values(101,'Arun'60);

Insert into student values(101,'Arun'60);

Insert into student values(102,'Arun'70);

Insert into student values(102,'Arun'70);

Select * from student;

Delete from student where Rownum IN(1,2,3); // this is not right query

Select Rowid, sno, sname, marks from student;

Select min(Rowid) from student;

Select min(Rowid) from student group by sno;

***Subqueries:**

Subqueries are used to get the result based on unknown values. They are different type.

1. Single Row subquery
2. Multiple Row subquery
3. Multiple column subquery
4. Co-related subquery
5. Scalar subquery
6. Inline view

***Single Row Subquery:**

When subquery returns one row (1 value). It is called Single RowSubquery.

Ex: write a query to display details are having salary > 'ALLENS' sal ?

Select * from emp where sal > (select sal from emp where ename = 'ALLEN');
o/p: 1600

Note:

Subqueries should always place in the inside.

Subqueries are executed first and then parent query is executed by using the result of sub query.

Level Two query:

Select * from emp where job = (select job from emp where ename = 'ALLEN')
AND job = (select job from emp where ename = 'BLAKE');

Level Three query:

Select * from emp where sal > (select sal from emp
Where ename = (select ename from emp
Where empno = 7499));

Note: The above query is three level query.

Sub query can be nested upto 32 levels.

****Multiple Row Subquery:**

When subquery returns multiple rows. It is called multiple row salary.

Note: we should use multiple row operators with multiple row subqueries. They are three multiple row operators.

1. IN
2. ANY
3. ALL

***ALL:** Select * from emp

Where sal > ALL(Select sal from emp
Where deptno = 30);

<u>Empno</u>	<u>ename</u>	<u>job</u>	<u>sal</u>
7369	SMITH	salesman	2975
7860	ALLEAN	ANALYST	3000

Ex: Select * from emp where sal < ALL(1600,2500,1250,3000,950);

*ANY: Select * from emp where sal > ANY(select sal from emp
where deptno = 30);

Select * from emp where sal < ANY(select sal from emp where deptno = 30);

*IN: Select * from emp where ename IN('ALLEN', 'KING', 'FORD');

Select * from emp where sal IN(select sal from emp where deptno = 30);

MULTIPLE COLUMN SUBQUERY:

When subquery return more then one column. It is called multiple column subquery.

We should use in operator with multiple column subqueries.

Ex:

Select * from emp where(job,sal) IN(select job, sal from emp where deptno = 30);

o/p: Job sal
 salesman 1600
 manager 1250
 salesman 2850

Note: In the o/p we get the rows when both the values are matching.

Delete some values:

Select * from student;

Select min(rowid) from student group by sno;

Select max(rowid) from student group by sno;

Delete from student

 where rowid not
 IN(select min(rowid) from
 student group by sno);

*write a query to display the row from emp table who is having the highest salary?

Select * from emp where sal = (select max(sal) from emp);

*write a query to display all the rows who are having salary grater than AVG
salary of emp table?

Select * from emp where sal >(select AVG(sal) from emp);

*write a query to display all deptno AVG salary?

Select deptno, AVG(sal) from emp group by deptno;

***Co-RELATED SUBQUERY:**

When subquery is executed in relation to parent query, it is called co-related subquery.

*write a query to display all the rows who are having salary grater than AVG salary his department?

Select AVG(sal) from emp;

Select * from emp where sal > (select AVG(sal) from emp group by deptno); //invalid

Select * from emp where sal > (select AVG(sal) from emp where deptno = 10);

***Select * from emp e

where sal > (select AVG(sal) from emp where deptno = e.deptno);

o/p: sal deptno

1600	30
2975	20
2850	30
3000	20
5000	10
3000	20

The above example is a co-related subquery.

In co-related subquery, parent query is executed first and then subquery is executed in relation to result of parent query.

***SCALAR subquery:** when we use subquery in the select clause. It is called as Scalar subquery.

*write a query to display following output?

<u>Deptno</u>	<u>Dname</u>	<u>loc</u>	<u>sumsal</u>
10	Accounting	New York	8750
20	Research	Dallas	10875
30	Sales	Chicago	9400
40	Operations	Boston	-----

```
Select deptno, dname, loc, (Select sum(sal) from emp where deptno = d.deptno)
      Sum_sal from dept d;
```

Scalar subquery are also called sub select.

***INLINE VIEW:**

When a subquery is used in from clause. It is called INLINE view.

```
Select Rownum, empno, ename, sal, deptno from emp;
```

```
Select * from emp where Rownum <=5;
```

```
Select * from emp;
```

```
Select * from emp ORDER BY sal desc;
```

*write a query to display details of employees who are having top 5 salaries?

```
Select * from emp where Rownum <=5 ORDER BY sal desc;
```

```
Select * from (select * from emp ORDER BY sal desc) where rownum <=5;
```

*write a query to display details of 5th highest salary?

```
Select * from (select * from emp ORDER BY sal desc)
               where rownum <=5)
```

minus

```
Select * from (select * from emp ORDER BY sal desc)
               where rownum <=4;
```

<u>clause</u>	<u>subcaluse</u>
select yes	scalar
from yes	inline
where yes	
group by no	
having yes	
order by no	

INTERVIEW QUESTIONS

1. What are pseudo columns?

It is rownum is a pseudo column which starts with one and increment by 1.

2. Write a query to display first n rows from the table?

Select rownum, empno, ename, sal, deptno from emp;

3. What are different between rownum and rowid?

Rownum

Rownum values starts with 1 and increment by one.

Rownum values are temporary.

Rownum values are generated when query is executed.

Rowid

Rowid's are hexadecimal values.

Rowid values are permanent.

The Rowid values are generated when row is created or inserted.

4. Write query to delete duplicate rows from the table?

Delete from student where Rowid Not IN(select min(Rowid) from student group by sno);

5. write a query to display the first five of highest?

Select * from(select * from emp ORDER BY sal desc) where rownum <=5)

Minus

Select * from(select * from emp ORDER BY sal desc) where rownum <=4);

6. Explain about correlated subquery?

When subquery is executed in relation to parent query, it is called correlated subquery.

7. Whar are multiple row operators?

IN

ANY

ALL

8. Explain scalar subquery?

When we use sub query in the select clause it is called scalar subquery.

9. Explain inline view?

When a sub query is used inform clause it is called inline view.

***views**: view is a logical representation of data from one or more then one table.

They are different types

Simple views

- Complex views
- Read only views
- With check option views
- Materialized views

***Single views:** when view is created using one base table it is called as Single view.

Syntax:

Create view<view_name> as <select STMT>;

Ex:

Create view v1 as select empno, ename, sal from emp;

View does not contain any data.

View does not consume memory location.

When we write select statement on view, we get the data from the table.

Ex: Select * from v1;

<u>Empno</u>	<u>Ename</u>	<u>Sal</u>
7369	Smith	800
7499	Allen	1600
-----	-----	-----
-----	-----	-----

Tables which are used for creating the view are called as above tables.

Select * from TAB; will give list of all the tables and view which are the data base tables.

Ex:

Create view emp_V10 AS select empno, ename, sal, deptno, from emp

where deptno = 10;

Create view emp_V10 AS select empno, ename, sal, deptno, from emp

where deptno = 20;

Create view emp_V10 AS select empno, ename, sal, deptno, from emp

where deptno = 30;

Select * from V10;

Select * from V20;

Select * from V30;

We can perform DML operations on simple views.

Any DML operation performed on simple view will be reflected on base table.

To see the structure of the table.

Ex:

DESC V1

Name	Null	Type
Empno	Notnull	number(4)
Ename		varchar2(10)
Sal		number(7,2)

*Query to see only view tables?

Select view_name from user_views;

Outputs: V1, emp_v10, emp_20, emp_30, V5

user_views is an example of data dictionary table.

Create view test_v6 as select empno, ename, sal, deptno from emp where deptno = 30;

Select * from Test_v6; // 6 rows are selected

Insert into Test_v6 values(6868, 'Anil', 2000, 10);

Select * from Test_v6; // 6 rows are selected

Select empno, ename, sal, deptno from emp where deptno = 30;

<u>View_name</u>	<u>Text</u>
Test_v6	select empno, ename, sal, deptno from emp where deptno = 30;

In user_view from the database we get list of all the view and corresponding select statement used for the view.

Select view_name, Text from user_views;

***Complex view:**

When a view is created using multiple base tables it is called Complex view.

Ex:

Create view Test_v7

As select e.empno, e.ename, e.sal, e.deptno,
d.dname, d.loc from emp e, dept d where e.deptno = d.deptno;

insert into Test_v7 values(7878, 'ravi', 9000, 40, 'HR', 'HYD'); // Error

msg: DML operations are not allowed in complex views.

Create view Test_v8

As select empno, ename, sal, sal*12 from emp; // Error

Create view Test_v8

As select empno, ename, sal, sal*12 Annual_sal from emp;

Select * from Test_v8;

Insert into Test_v8 values(1212,'GMR',1000,12000); // Error

Create view Test_v9

As select empno, ename, Lower(ename) Name from emp;

A view is called as complex view when we use arithmetic operations and functions or group by clauses.

Create view Test_v10

As select deptno, sum(sal) sum_sal from emp group by deptno;

***Different between simple and complex views?

Simple view

1. Created by using only one table.
2. DML operations are allowed.
3. Should not be created using arithmetic operations or functions or group by clauses.

Complex view

1. Created by using multiple tables.
2. DML operations are not allowed.
3. Can be created using arithmetic operations or functions or group by clauses.

***Read Only View:**

We can restrict DML operation views by creating read only view.

Ex:

Create view v3

As select empno, ename, sal, deptno from emp with read only;

Select * from v3;

Insert into v3 values(3131,'ABC',10000,60); // Error

Create or replace clause is used to change definition of the view.

Create or replace view v4

As select empno, ename, sal, deptno, job from emp;

***With Check Option View:**

These views will allow DML operation only when where condition is satisfied.

Create view Test_V12

As select empno, ename, sal, deptno from emp

Where deptno = 30

With check option;

Insert into Test_V12 values(666,'AAA',4000,30); // valid

Insert into Test_V12 values(888,'AAA',4000,10); // error (invalid)

Create view Test_V13

As select empno, ename, sal, deptno from emp

Where sal > 2000

With check option;

Select * from Test_V13;

Insert into Test_V13 values(6969,'AAA',3100,10); // valid

Insert into Test_V13 values(6955,'AAA',1510,10); // Invalid

Update Test_V13 set sal = 4000 where empno = 7566; //valid

Update Test_V13 set sal = 1100 where empno = 7902; // invalid

*materialized view:

Materialized views will help improving the performance of select statement on view.

To create materialized view, the based table should have primary key.

Changes to base table will not reflect on materialized view.

Materialized view or previously called as snap short.

Ex:

Create view Test_V14

As select empno, ename, sal, deptno from emp;

Create view Test_V15

As select e.empno, e.ename, e.sal, e.deptno, d.dname, d.loc

from emp e, dept d where e.deptno = d.deptno;

select * from Test_V14; //performance fast

select * from Test_V15; //performance fast two tables

Syntax:

Create MATERIALIZED view <VIEW_NAME> AS <select STMT>;

Ex:

Create materialized view MV1
As select empno, ename, sal, deptno from emp;
Select * from MV1;

***To Refresh materialized View:**

Sql> exec DBMS_SNAPSHOT.REFRESH('MV1');
Select * from MV1;

DBMS_SNAPSHOT-PACKAGE NAME
REFRESH ---procedures

Select view_name from user_views; // All view tables are display

***To Drop a view:**

Syntax:

Drop view <view_name>;

Ex:

Drop view Test_V14;

When base tables are drop, the view becomes invalid.

When base tables are re_created view will become valid.

*****INDEX:**

Index is an object which is used to improve the performance of select statements.

Types of Indexes: They are two types of Indexes.

1. Simple Index
2. Composite Index

1.Simple Index:

When index is created on one column it is called as simple index.

Syntax:

CREATE INDEX <INDEX_NAME> ON <TABLE_NAME> (COL_NAME);

Ex:

Create index IDX1 on emp(sal);

Index should be created on columns which we regularly use in the where clause.

When a index is created a separate structure is created with first column is ROWID and second column as indexed column.

The Rows in the index will be arranged in the ascending order of indexed column.

IDX1

ROWID	SAL
	800
	950
	1100
	1250
	1600
	5000

Using algorithm is identifies the back of ROWID's
Using which rows are displayed.

***Composite Index:** when Index is created multiple columns it is called composite index.

Ex: create index IDX2 on emp(sal,job);

The above index IDX2 is used only when both the columns are used in the where clause.

Disadvantages of index:

Index will consume memory.

The performance of DML command will be decreased.

Index can also be categorized two types:

1. Unique index
2. Non-unique index

***Unique Index:**

If the indexed column contains unique value it is called unique index.

A unique index is automatically created. When we create a table with primary key constraint or unique constraint.

***Non-unique index:**

If an index column contains duplicated values they are called as non-unique index.

Ex:

Create index IDX1 on emp(sal);

See to index tables:

Select index_name, from user_indexes;

*Query to see list of all the indexes.

Select index_name, table_name from user_indexes;

*Query to see list of all the indexes along with column name.

Select index_name, table_name, column_name from user_ind_columns;

Desc user_indexes

Desc user_ind_columns

***function based index:**

When index is created by using functions it is called as function based index.

Ex:

Create index indexs on emp (lower(ename));

The index is used only when use the appropriate function in the where clause function.

Select * from emp where lower(ename) = 'king';

To drop on index:

Ex:

Drop INDEX IDX1;

***Sequences:** sequence is an object which is used to generate numbers.

Syn:

Create sequence <SEQUENCE_NAME> start with <value> increment by <value>;

Ex:

Create sequence SE1 start with 1 increment by 1;

Ex:

Create sequence SE4 start with 1000 increment by 1 maxvalue 5000 cycle;

***Using the Sequence:** There are two pseudo to sequence.

1. NEXTVAL

2. CURRVAL

***NEXTVAL:** Nextval is used to generate a number.

***CURRVAL:** Currvval is used to know the latest number generated.

Create sequence SE5 start with 101 increment by 1;

Insert into student7 values(SE5.NEXTVAL,'Arun',60);

Insert into student7 values(SE5.NEXTVAL,'Amit'61);

Sequence is a sharable object.

When sequence is shared we get gaps in numbers.

Example of CURRVAL: To know the latest value generated.

Ex:

Select SE5.CURRVAL from dual;

Sequence with cache option will help in generating the numbers faster.

Ex:

Create sequence SE6 start with 1000 increment by 1 maxvalue 10000 cycle cache 40;

Cache option will help in improving the performance of sequence.

***Synonym:** it is an alternate name given to an object.

Syntax: create synonym <Synonym_name> for <Table_name>;

Ex:

Create synonym E1 for emp;

Synonym helps in reducing the length of the query.

Synonym is used instead of table names for all the commands.

Ex:

Select * from E1;

Query to see all synonyms:

SQL> select synonym_name from user_synonyms;

To drop a synonym:

SQL> DROP SYNONYM E1;

Object of Oracle:

1. Table
2. View
3. Index

4. Sequences
5. Synonyms

Synonym Objects of PL/SQL:

1. PROCEDURE
2. FUNCTION
3. PACKAGE
4. TRIGGER

Normalization: Normalization is process of removing redundancy and improving accuracy of the database. Normalization can be achieved by using normal forms.

***1st Normal form (1NF)**: A database is in 1NF if it satisfies following rules.

Rule1: Each cell should have one value.

Rule2: Table should have primary key.

Ex:

Author ID	Author Name	Book Name	Book Cost	PRIMARY KEY
A101	IVAN	SQL	200/-	Author ID
A101	IVAN	PLSQL	250/-	Author Name
A102	RAGHU	JAVA	150/-	
A102	RAGHU	J2EE	250	

***2nd Second Normal form (2NF)**: A database is in 2NF if it satisfies following rules.

Rule1: Database should be in 1NF.

Rule2: There should be no partial dependency.

Partial dependency: When a non-key attribute is dependent on part of a primary key.

Then there exists partial dependency. The following table is satisfying 2NF rules.

The diagram illustrates a partial dependency in a table. A primary key arrow originates from the 'Supplier ID' column and points to the 'Supplier ID' column. A partial dependency arrow originates from the 'S Name' column and points to the 'Supplier ID' column, indicating that the 'S Name' attribute depends on part of the primary key ('Supplier ID').

Part ID	Supplier ID	S Name	Price	Address
65	2	TATA	59	Bangalore
73	2	TATA	60	Bangalore
65	1	BIRLA	54	Hyderabad

In the above table S Name (non key attribute) is depending on supplier ID (part of

primary key). Hence there exists partial dependency. We can eliminate partial dependency by dividing the table into two different tables. The following tables are satisfying 2NF rules.

Primary key		
Part ID	Supplier ID	Price
65	2	59
73	2	60
65	1	54

Primary key		
Supplier ID	S Name	Address
2	TATA	Bangalore
1	BIRLA	Hyderabad

***3rd Normal form (3NF)**: A database is in 3NF if it satisfies the following rules.

Rule1: Database should be in 2nd NF.

Rule2: There should be no transitive dependency.

Transitive dependency: When a non key attribute is dependent on another non key attribute then there exists transitive dependency the following table is not satisfying 3rd NF rules.

PART NO	MANFNAME	MANFADDRESS
1000	TOYOTA	PARK AVENUE
1001	MISTUBUSHI	LOS ANGELS
1002	TOYOTA	PARK AVENUE

In the above table manufacture address (non key) is dependent of manufacture name (non key). Hence there exists transitive dependency.

We can eliminate transitive dependencies by dividing the table into two different tables.

Primary key	
Part no	ManufName
1000	Toyota
1001	Mistubishi
1002	Toyota

ManfName	ManfAddress
Toyota	Park Avenue
Mistubishi	Los Angels

Interview Questions:

- What is view?

A view is a logical representation of data from one or more than one table.

- List of differences between simple views and complex views?

3. In which cases a view is called complex view?

When a view is created using multiple base tables it is called complex view.

4. How can we restrict DML operations on simple views?

We can perform DML operations on simple views. Any DML operation performance on simple view will be reflected on base table.

5. What is with check option view?

These views will allow DML operation only when where condition is satisfied.

6. Do you think view contains data?

View does not contain any data.

View does not consume memory location.

When we write select statement on view, we get the data from the table.

7. What is a data dictionary table used to see the list of view?

8. What is a materialized view?

Materialized views will help improving the performance of select statement on view.

9. What is the advantage of index?

Index is an object which is used to improve the performance of select statements.

10.What is the disadvantage of index?

Index will consume memory.

The performance of DML command will be decreased.

11.What is unique index?

If the indexed column contains unique value it is called unique index.

A unique index is automatically created. When we create a table with primary key constraint or unique constraint.

12.What is sequence?

Sequence is an object is used to generate numbers.

13.What is different between simple index and composite index?

14.What are pseudo columns related to sequence?

15. When can we have gaps in sequence?

16. What is a synonym?

It is an alternate name given to an object.

17. What is different between table alias and synonym?

CODD RULES: These rules are developed by Mr.'s E F CODD.

If a DBMS satisfies at least 6 rules out of 12 then it is called as RDBMS.

Rule 1: The information rule:

All information in the data type is to be represented in one and only one way, in the form of rows and columns.

Rule 2: The guaranteed access rule:

If you can insert, you should be able to select.

Rule 3: Systematic treatment of null values:

The DBMS must allow each field to remain null (or empty) specifically, It must support a representation of missing information and inapplicable information.

Rule 4: Achieve online catalog based on the relational model:

Users must be able to access the database's structure (catalog) using the same query language that they use to access the database's data.

Rule 5: The comprehensive data sub language rule:

Language should be divided into several sub languages based on activities the command will perform.

Rule 6: The view updating rule:

All views that are theoretically updatable must be updateable by the system.

Rule 7: High-level insert, update, and delete:

This rule states that insert, update and delete operations should be supported for any retrievable set rather than just for a single row in a single table.

Rule 8: Physical data independence:

Changes to the physical level (How the data is stored, whether in arrays or linked lists etc.) must not require a change on application based on the structure.

Rule 9: Logical data independence:

Changes to the logical level (tables, columns, rows, and so on) must not require a change to an application based on the structure.

Rule 10: Integrity independence:

This rule says that, it must be possible to change such constraints as and when appropriate without unnecessarily affecting existing application.

Rule 11: Distribution independence:

The distribution of portions of the database to various locations should be invisible to users of the database.

Rule 12: The non subversion rule:

If the system provides a low-level (record-at-a-time) interface, then that interface cannot be used to subvert (degrade) the system.

***Script:** A script is a file which is collection of commands same with extension .SQL

To run the script @D:\malli\FIRST.SQL

*****PL/SQL*****

It is an extension of SQL. The following are advantages of PL/SQL.

1. We can use programming features like if statement loops etc.
2. PL/SQL helps in reducing network traffic.
3. We can have user defined error messages by using concept of exception handling.
4. We can perform related actions by using concept of Triggers.
5. We can save the source code permanently for repeated execution.

PL/SQL Block:

A PL/SQL program is called as PL/SQL block.

PL/SQL Block:

DECLARE

```
-----  
----- --DECLARE SECTION --OPTIONAL  
  
-----  
BEGIN  
  
-----  
----- --EXECUTABLE SECTION --MANDATORY  
  
-----  
EXCEPTION  
  
-----  
----- --EXCEPTION SECTION --OPTIONAL  
  
-----  
END;  
/
```

***Declare:** This section is used to declare local variables, cursors, Exceptions and etc. This section is optional.

***Executable Section:** This section contains lines of code which is used to complete table. It is mandatory.

***Exception Section:** This section contains lines of code which will be executed only when exception is raised. This section is optional.

***Simplest PL/SQL Block:**

```
Begin  
-----  
-----  
-----  
END;  
/
```

*write a PL/SQL Block which will display Hello World?

```
SET SERVEROUTPUT ON
```

```
Begin  
DBMS_OUTPUT.PUT_LINE( 'HELLO WORLD' );
```

```
END;  
/  
  
o/p: Hello world  
PL/SQL procedure successfully completed.
```

Note: To get the output of the program server output environment variable should be on.

Command: SET SERVEROUTPUT ON

DBMS_OUTPUT is name of the PACKAGE

.PUT_LINE is name of the PROCEDURE

'/' Slash is used to submit the program in DBS.

*Write PL/SQL block which will calculate sum of two numbers and display the output?

```
DECLARE  
A number(2);  
B number(2);  
C number(3);  
BEGIN  
A := 10;  
B := 20;  
C := A + B;  
DBMS_OUTPUT.PUT_LINE(C);  
DBMS_OUTPUT.PUT_LINE('sum of two numbers' || C);  
END;  
/
```

```
o/p: 30  
o/p: sum of two numbers 30  
PL/SQL procedure successfully completed.
```

--is a Single line comment.

/* */ ----Multi-Line comment

Ex:

```
Initialization in declare section  
A number(2) := 50;  
B number(2) := 25;
```

*Write a PL/SQL block which accepts two numbers from the user and display the sum?

```
DECLARE
A number(2);
B number(2);
C number(3);
BEGIN
A := &A; or A := &malli;
B := &B; or B := &iNetSlov
C := A + B;
DBMS_OUTPUT.PUT_LINE( 'sum of the two numbers' || C);
END;
/
```

To modify the data

```
Begin
Merge-----
Insert-----
Update-----
Commit-----
Select----- // not execute
END;
/
DCL-----NO
DDL-----NO
DML-----YES
DRL-----NO
TCL-----YES
```

*Write a PL/SQL block which accepts employee number and increment is salary by 1000?

```
DECLARE
A number(4);
A := &Empno;
Update emp set sal = sal + 1000 where Empno = A;
END;
```

/

*Write a PL/SQL block which empno and delete that row from the emp table?

```
DECLARE
A number(4);
BEGIN
A := &Empno;
Delete from emp where Empno = A;
END;
/
```

*Control statement: If – Then – Else

```
Declare
A number := 5;
Begin
DBMS_OUTPUT.PUT_LINE( 'welcome' );
If A > 10 Then
DBMS_OUTPUT.PUT_LINE( 'HELLO1' );
Else
DBMS_OUTPUT.PUT_LINE( 'HELLO2' );
END If;
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
END;
/
```

*LOOPS: There are three types

1. Simple loop
2. While loop
3. for loop

1. Simple:

```
Declare
A number(2) := 1;
Begin
DBMS_OUTPUT.PUT_LINE( 'welcome' );
LOOP
```

```
DBMS_OUTPUT.PUT_LINE( 'HELLO1' );
DBMS_OUTPUT.PUT_LINE( 'HELLO2' );
Exit when A = 4;
A := A + 1;
END LOOP;
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
END;
/
```

2. **While:**

```
Declare
A number(2) :=1;
Begin
DBMS_OUTPUT.PUT_LINE( 'WELCOME' );
While A <=4 loop
DBMS_OUTPUT.PUT_LINE( 'HELLO1' );
DBMS_OUTPUT.PUT_LINE( 'HELLO2' );
A := A + 1;
END LOOP;
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
END;
/
```

3. **FOR LOOP:**

```
Declare
A number;
Begin
DBMS_OUTPUT.PUT_LINE( 'WELCOME' );
FOR A IN 1 .. 4 LOOP
DBMS_OUTPUT.PUT_LINE( 'HELLO1' );
DBMS_OUTPUT.PUT_LINE( 'HELLO2' );
END LOOP;
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
END;
/
```

Note: in for loop the loop variable is implicitly declare.

*write a select statement in PL/SQL block?

Every select statement in a PL/SQL block should have into clause.

***write a program to display employee name for the empno 7698?

Declare

```
A varchar2(15); // Ename varchar2(15);
```

Begin

```
Select ename into A from emp where empno = 7698;
```

```
DBMS_OUTPUT.PUT_LINE( A );
```

```
END;
```

```
/
```

*write a program accepts deptno and display deptname and location?

Declare

```
L_Deptno number(2);
```

```
L_DName varchar2(15);
```

```
L_Loc varchar2(10);
```

Begin

```
L_Deptno := &Deptno;
```

```
Select DName, Loc into L_DName, L_Loc from dept where Deptno = L_Deptno;
```

```
DBMS_OUTPUT.PUT_LINE( 'L-DName' );
```

```
DBMS_OUTPUT.PUT_LINE( 'L_Loc' );
```

```
END;
```

```
/
```

***Using % type attribute:**

Percentage type attribute is used to declare local variable with respect to column of a table.

Syntax: <VAR_NAME><TABLE_NAME>.<COL_NAME> % TYPE;

Ex:

```
L_Dname Dept.Dname%TYPE;
```

*Write a program which accepts empno to display ename, job and salary?

Declare

```
L_empno emp.empno % type;
```

```
L_ename emp.ename % type;
```

```
L_job emp.job % type;
```

```
L_salary emp.salary % type;
```

Begin

```
L_empno := &empno;  
Select ename, job, salary into L_ename, L_job, L_salary from emp where empno = L_empno;  
DBMS_OUTPUT.PUT_LINE( 'L_ename' );  
DBMS_OUTPUT.PUT_LINE( 'L_job' );  
DBMS_OUTPUT.PUT_LINE( 'L_salary' );
```

(or)

```
DBMS_OUTPUT.PUT_LINE( L_ename || ... || L_job || ... || L_salary);  
END;  
/
```

Note:

When a select statement does not return any row program will terminate abnormally.
When select statement returns more than one row, program will terminate abnormally.

***Percentage (%) Row type attributes:**

This attribute to declare a local variable which can store complete Row of a table.

Syn:

```
<VARIABLE_NAME><TABLE_NAME>% ROW TYPE;
```

Ex:

```
A EMP%ROWTYPE;
```

We cannot directly display a Row type variable.

We can access one value in Row type variable by using.

```
<VARIABLE_NAME>.<COL_NAME>
```

Ex:

Declare

```
A EMP%ROWTYPE;
```

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE( 'WELCOME' );
```

```
Select * INTO A from emp where empno = 7782;
```

```
DBMS_OUTPUT.PUT_LINE(A.sal || A.job || A.Hiredate);
```

```
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
```

```
END;
```

```
/
```

***EXCEPTIONS:**

Runtime Errors are called as Exceptions. They are three types of Exceptions.

1. ORACLE Predefined Exception
2. ORACLE Non Predefined Exception
3. USER Defined Exception

***Oracle Predefined Exception:**

These Exceptions will have Exception name and Exception number. Examples of predefined Exceptions are:

1. NO_DATA_FOUND
2. TOO_MANY_ROWS
3. ZERO_DIVIDE
4. VALUE_ERROR
5. DUP_VAL_ON_INDEX

***NO_DATA_FOUND:** This Exception is raised when select statement does not return any Row.

Ex:

```
Declare
  L_sal emp.sal%type;
Begin
  DBMS_OUTPUT.PUT_LINE( 'WELCOME' );
  Select sal INTO L_sal from emp where empno = &empno;
  DBMS_OUTPUT.PUT_LINE(L_sal);
  DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
EXCEPTION
  when NO_DATA_FOUND then
    DBMS_OUTPUT.PUT_LINE( 'INVALID EMPNO' );
  END;
  /
```

***TOO_MANY_ROWS:**

This Exception is raised when select statement more then one row.

Ex:

```
Declare
  L_sal emp.sal%type;
Begin
  DBMS_OUTPUT.PUT_LINE( 'WELCOME' );
  Select sal INTO L_sal from emp where deptno = 30;
```

```
DBMS_OUTPUT.PUT_LINE(L_sal);
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
EXCEPTION
when TOO_MANY_ROWS then
DBMS_OUTPUT.PUT_LINE( 'MORE THEN ONE ROW RETURNED' );
END;
/
```

*ZERO_DIVIDE:

Ex:

```
Declare
A Number;
Begin
A := 5/0;
Exception
when ZERO_DIVIDE then
DBMS_OUTPUT.PUT_LINE( 'DO NOT DIVIDE BY 0' );
END;
/
```

Note:

This Exception is raised when we try to divided by zero.

***VALUE_ERROR:** This Exception is raised when there is miss match with the value and data type of local variable or size of local variables.

Ex 1:

```
Declare
L_sal emp.sal%type;
Begin
DBMS_OUTPUT.PUT_LINE( 'WELCOME' );
Select ename INTO L_sal from emp where empno = 7521;
DBMS_OUTPUT.PUT_LINE(L_sal);
DBMS_OUTPUT.PUT_LINE( 'THANK YOU' );
EXCEPTION
when VALUE_ERROR then
DBMS_OUTPUT.PUT_LINE( 'please check the local variables' );
END;
/
```

Ex 2:

```
Declare
A number(3);
Begin
A := 1234;
Exception
when VALUE_ERROR then
DBMS_OUTPUT.PUT_LINE( 'PLEASE CHECK THE LOCAL VARIABLES' );
END;
/
```

***DUP_VAL_ON_INDEX:** (duplicate value on index)

This Exception is raised when we try to insert a duplicate value in primary key constraint.

Ex:

```
Begin
DBMS_OUTPUT.PUT_LINE( 'welcome' );
Insert into student values(104, 'ARUN',60);
DBMS_OUTPUT.PUT_LINE( 'Thank you' );
Exception
when DUP_VAL_ON_INDEX then
DBMS_OUTPUT.PUT_LINE(' Do not insert duplicates' );
END;
/
```

The above program works on an assumption the table student for if having a primary key SNO column with value 104.

<u>Exception Name</u>	<u>Exception No</u>	<u>Exception Message</u>
1.NO_DATA_FOUND	ORA-1403	NO DATA FOUND
2.TOO_MANY_ROWS	ORA-1422	EXACT FETCHED REQUESTED MORE THEN ONE ROW
3.ZERO_DIVIDE	ORA-1476	DIVISON IS EQUAL TO ZERO
4.VALUE_ERROR	ORA-6502	NUMRIC OR VALUE ERROR
5.DUP_VAL_ON_INDEX	ORA-0001	UNIQUE CONSTRAINT VIOLATED

***WHEN OTHERS:**

When others are a universal Exception angular this can catch all the Exceptions.

```
Declare
L_sal number(4);
A number;
Begin
DBMS_OUTPUT.PUT_LINE( 'Welcome' );
Select sal INTO L_SAL from emp where deptno = &deptno;
DBMS_OUTPUT.PUT_LINE('The sal is ....'||L_sal);
A :=10/0;
DBMS_OUTPUT.PUT_LINE( 'Thank you' );
Exception
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE( 'please check the code' );
END;
/
```

***ERROR REPORTING FUNCTIONS:** They are two Error Reporting functions.

1. SQLCODE
2. SQLERRM

These error reporting functions are used in when others clause to identified the exception which is raised.

1. **SQLCODE**: It returns ERRORCODE
2. **SQLERRM**: It returns Exception number and Exception message.

Note: for NO_DATA_FOUND Exception SQLCODE will return 100.

```
Declare
L_sal number(4);
A number;
Begin
DBMS_OUTPUT.PUT_LINE( 'Welcome' );
Select sal INTO L_SAL from emp where deptno = &deptno;
DBMS_OUTPUT.PUT_LINE('The sal is ....'||L_sal);
A :=15/0;
DBMS_OUTPUT.PUT_LINE( 'Thank you' );
```

```
Exception
WHEN OTHERS THEN
DBMS_OUTPUT.PUT_LINE( 'please check the code' );
DBMS_OUTPUT.PUT_LINE(SQLCODE);
DBMS_OUTPUT.PUT_LINE(SQLERRM);
END;
/
```

*NESTED BLOCK:

```
Declare
A number := 10;
Begin
DBMS_OUTPUT.PUT_LINE('HELLO1');
Declare
B number := 20;
Begin
DBMS_OUTPUT.PUT_LINE('HELLO2');
DBMS_OUTPUT.PUT_LINE(B);
DBMS_OUTPUT.PUT_LINE(A);
END;
DBMS_OUTPUT.PUT_LINE('HELLO3');
DBMS_OUTPUT.PUT_LINE(B); --ERROR
END;
/
```

Note: outer block variables can be accessed in nested block nested block variables can not be accessed in outer block.

*EXCEPTION PROPAGATION:

```
Begin
DBMS_OUTPUT.PUT_LINE('HELLO1');
L_SAL EMP.SAL%TYPE;
Begin
DBMS_OUTPUT.PUT_LINE('HELLO2');
Select sal INTO L_SAL from emp where empno = 1111;
DBMS_OUTPUT.PUT_LINE('HELLO3');
END;
DBMS_OUTPUT.PUT_LINE('HELLO4');
```

```
EXCEPTION  
WHEN NO_DATA_FOUND THEN  
DBMS_OUTPUT.PUT_LINE('HELLO5');  
END;  
/
```

***ORACLE NON PREDEFINED EXCEPTIONS:**

These Exceptions will have only Exception number. But does not have Exception name.
Steps to handle non predefined exceptions.

Syntax:

Step1: Declare the Exception

```
<EXCEPTION_NAME> EXCEPTION;
```

Step2: Associate the Exception

```
PRAGMA EXCEPTION_INIT(<EXCEPTION_NAME>, <EXCEPTION NO>);
```

Step3: Catch the Exception

```
WHEN <EXCEPTION_NAME> THEN
```

```
-----  
-----  
-----
```

```
END;
```

```
/
```

ORA -2292 is an example of non predefined exception.

This exception is raised when we delete row from a parent table. If the corresponding value existing the child table.

Declare

```
MY_EX1 Exception; --step1
```

```
PRAGMA EXCEPTION_INIT(MY_EX1, -2292); --step2
```

Begin

```
DBMS_OUTPUT.PUT_LINE('Welcome');
```

```
Select from student where eno = 102;
```

EXCEPTION

```
WHEN MY_EX1 THEN --step3
```

```
DBMS_OUTPUT.PUT_LINE('do not delete paragma table');
```

```
END;
```

/

Pragma Exception_init is a compiler directive which is used to associate an Exception name to the predefined number.

***USER DEFINED EXCEPTION:**

These Exceptions are defined and controlled by the user. These Exceptions neither have predefined name nor have predefined number. Steps to handle user defined Exceptions.

Step1: Declare the Exception

Step2: Raised the Exception

Step3: Catch the Exception

Declare

MY_EX1 EXCEPTION; --Step1

L_SAL EMP.SAL%TYPE;

Begin

DBMS_OUTPUT.PUT_LINE('welcome');

Select SAL INTO L_SAL from emp where empno = &empno;

IF L_SAL > 2000 THEN

RAISE MY_EX1; --Step2

ENDIF;

DBMS_OUTPUT.PUT_LINE('The sal is ... '|L_sal);

DBMS_OUTPUT.PUT_LINE('Thank you');

EXCEPTION

WHEN MY_EX1 THEN --Step3

DBMS_OUTPUT.PUT_LINE('Sal is too high');

END;

/

Note: When others should be the last handler of the exception section otherwise we get a compiler ERROR.

***RAISE_APPLICATION_ERROR:** RAISE_APPLICATION_ERROR is a procedure which is used to throw one error number and error message to the calling environment.

It internally performs a roll back.

ERROR number should be range of -20000 to -20999. ERROR message should be

displayed less than or equal to 512 characters.

```
Declare
L_sal emp.sal%TYPE;
Begin
DBMS_OUTPUT.PUT_LINE('Welcome');
Insert INTO dept values (08,'arun',70);
Select sal INTO L_sal from emp where empno = 7698;
IF L_sal > 2000 THEN
RAISE_APPLICATION_ERROR(-20150, 'SAL IS TOO HIGH');
END IF;
DBMS_OUTPUT.PUT_LINE('THE SAL IS...' || L_SAL);
END;
/
```

***CURSORS**: CURSOR is a memory location which is used to run SQL commands.

They are two types of cursors.

1. Implicit cursor
2. Explicit cursor

***Implicit cursor**: All the activities related to cursors like opening the cursor, processing the cursor and closing the cursor are done automatically. Hence these cursor are called as implicit cursor.

***Implicit cursor attributes**: They are 4 implicit cursor attributes.

1. SQL%ISOPEN
2. SQL%FOUND
3. SQL%NOTFOUND
4. SQL%ROWCOUNT

***SQL%ISOPEN**: It is a Boolean attribute. It always returns false.

***SQL%FOUND**: it is a Boolean. It returns true. If SQL command is successful returns false if SQL command is fails.

***SQL%NOTFOUND**: It is a Boolean attribute returns true. If SQL command is fails returns false if SQL command is successful.

Note:

SQL%NOTFOUND attribute is exactly opposite to SQL%FOUND.

***SQL%ROWCOUNT**: It returns the number of rows affected by SQL command.

***using SQL%FOUND**: It is used to know whether a specific command is effecting the table data or not.

Ex:

```
Begin
  Update emp set sal = 2000 where empno = 1111;
  IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('UPDATE SUCCESSFUL');
  ELSE
    DBMS_OUTPUT.PUT_LINE('UPDATE FAILED');
  END IF;
END;
/
```

***USING SQL%ROWCOUNT**:

```
Begin
  Update emp set sal = 2000 where deptno =30;
  DBMS_OUTPUT.PUT_LINE(SQL%ROWCOUNT||'ROWS UPDATED');
END;
/
```

SQL%ISOPEN

```
Begin
  Update emp set sal = 1000 where empno = 7654;
  IF SQL%ISOPEN THEN
    DBMS_OUTPUT.PUT_LINE('CURSOR IS OPEN');
  ELSE
    DBMS_OUTPUT.PUT_LINE('CURSOR IS CLOSED');
  END IF;
END;
/
```

Note: By the time we check whether cursor is open or not it is already closed by the oracle engine. Hence it is always returns false.

*write a PL/SQL block which accepts deptno and display all the employee names of the department?

Declare

L_ename emp.ename%TYPE;

Begin

Select ename into L_ename from emp where deptno = &deptno;

DBMS_OUTPUT.PUT_LINE(L_ename); --ERROR

END;

/

****EXPLICIT CURSORS:** All the activities related to cursor like.

1. Opening the cursor
2. Processing the cursor
3. Closing the cursor

Act should be done by the developer. Hence this cursor is called explicit cursors.

We should use explicit cursors to run a select statement. Which returns more than one row?

Steps to use explicit cursors:

Step 1: declare the cursor

Step 2: open the cursor

Step 3: fetch data from cursor to local variables

Step 4: close the cursor

Syntax:

Step 1: declare the cursor

CURSOR <CURSOR_NAME> IS <SELECT STMT>;

Step 2: open the cursor

OPEN <CURSOR_NAME>;

Step 3: fetch data from cursor to local variables

FETCH <CURSOR_NAME> INTO <VAR1>,<VAR2>,...,...,<VARn>;

Step 4: close the cursor

CLOSE <CURSOR_NAME>;

***EXPLICIT CURSOR ATTRIBUTES:** These are four explicit cursor attributes.

1. %ISOPEN
2. %FOUND
3. %NOTFOUND
4. %ROWCOUNT

***%ISOPEN:** It is a Boolean attribute which returns true when cursor is open.
Returns false when cursor is closed.

***%FOUND:** It is a Boolean attribute returns true when fetch statement is successful.

***%NOTFOUND:** It is a Boolean attribute returns true. When fetch statement fails.
Returns false when fetch statement successful.

Note: %NOTFOUND attributes is exactly opposite to % is found attribute.

***ROWCOUNT:** Returns number of rows processed by fetches statement.

*write a program to display all the employee names working in dept number 10?

Declare

Cursor c1

IS select ename from emp where deptno = 10;

L_ename emp.ename%TYPE;

Begin

open c1

loop

FETCH c1 into L_ename;

EXIT WHEN c1%NOTFOUND;

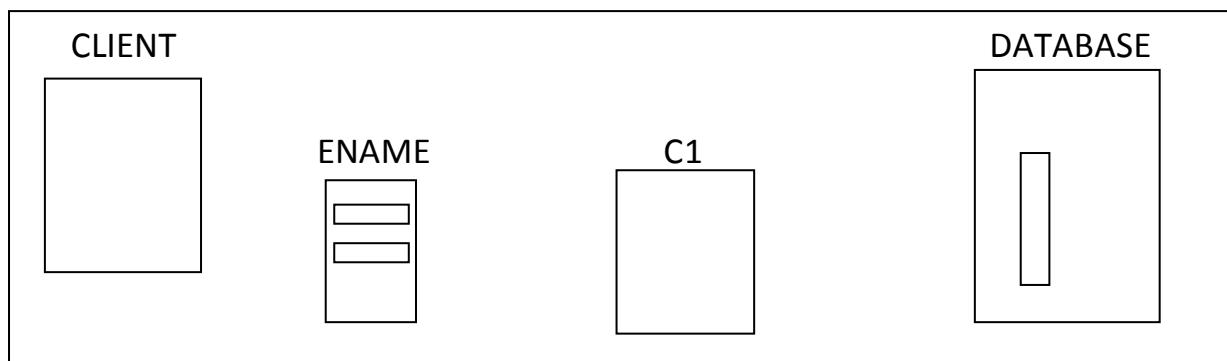
DBMS_OUTPUT.PUT_LINE(L_ename);

END LOOP;

CLOSE c1;

END;

/



ACTIVE DATA SET:

1. When we open a cursor the memory location is created.
2. Memory location will be loaded by data returned by select statement.
3. Cursor pointer points to 1st row of the memory location. This status is called active data set.

*write a PL/SQL block to display all the department names and locations from the department table?

```
Declare
Cursor c1
IS select dname, loc from dept;
L_dname dept.dname%TYPE;
L_loc dapt.loc%TYPE;
Begin
Open c1;
Loop
Fetch c1 into L_dname, L_loc;
EXIT WHEN c1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(L_dname||L_loc);
END loop;
Close c1;
END;
/
```

*write a program to display employee names, job, salary who are working in a department number 30?

```
Declare
CURSOR c1
IS select ename,job,salary from emp where deptno = 30;
A c1%ROWTYPE;
Begin
OPEN c1;
LOOP
FETCH c1 INTO A;
```

```
EXIT WHEN c1 %NOTFOUND
DBMS_OUTPUT.PUT_LINE(A.ename||'...'||A.job||'...'||A.salary);
END LOOP;
END;
/
```

***CURSOR FOR LOOPS:**

It is a short cut way of writing an explicit cursor program. When we use cursor for loop following steps are not required.

1. Opening the cursor not required.
2. Closing the cursor not required.
3. Fetch statement not required.
4. Exit when condition not required.
5. Declaring local variable not required.

```
Declare
CURSOR c1
IS select ename, job, salary from emp where deptno = 30;
Begin
FOR A IN c1 LOOP
DBMS_OUTPUT.PUT_LINE(A.ename||'...'||A.job||'...'||A.salary);
END LOOP;
END;
/
```

*Write a program to display the entire employee numbers their names and location?

```
Declare
CURSOR c1
IS select E.empno, E.ename, D.loc from emp E, Dept D where E.Deptno = D.Deptno;
Begin
FOR A IN c1 LOOP
DBMS_OUTPUT.PUT_LINE(A.empno||'...'||A.ename||'...'||A.loc);
END LOOP;
END;
/
```

SNO	SNAME	SUB1	SUB2	SUB3
-----	-------	------	------	------

101	ARUN	60	70	80
102	VARUN	65	77	80
103	SREENU	60	91	80
104	AMIT	60	70	88
105	VEERA	40	50	60

*Write a program it will calculate AVG marks from the entire student table?

```

Declare
CURSOR c1
IS select * from student;
L_AVG number(5.2);
Begin
FOR A IN c1 LOOP
L_AVG := (A.sub1 + A.sub2 + A.sub3)/3;
DBMS_OUTPUT.PUT_LINE('AVERAGE OF' || A.sno || 'IS' || L_AVG);
END LOOP;
END;
/

```

***Using%ISOPEN attribute:**

```

Declare
Cursor c1
IS select empno, ename from emp where deptno = 20;
L_empno emp.empno%TYPE;
L_ename emp.ename%TYPE;
Begin
open c1;
IF c1%ISOPEN then
DBMS_OUTPUT.PUT_LINE('cursor is open');
Else
DBMS_OUTPUT.PUT_LINE('cursor is closed');
END IF;
END;
/

```

***Using%ROWCOUNT attribute:** This attribute is used to fetch limited number of rows from the local variables.

Ex:

```
Declare
Cursor c1
IS select empno, ename from emp where deptno = 20;
L_empno emp.empno%TYPE;
L_ename emp.ename%TYPE;
Begin
open c1;
Loop
FETCH c1 INTO L_empno, L_ename;
Exit when c1%ROWCOUNT = 3;
DBMS_OUTPUT.PUT_LINE(L_empno||'....'||L_ename);
END Loop;
close(1);
END;
/
```

***Parameterized cursor:** A cursor which accepts a parameter from the user is called as parameterized cursor. Active dataset changes dynamically basing on the value passed to the cursor.

Ex:

```
Declare
Cursor c1(A number)
IS select empno, ename from emp where deptno = A;
L_empno emp.empno%TYPE;
L_ename emp.ename%TYPE;
Begin
open c1(&deptno);
Loop
FETCH c1 INTO L_empno,L_ename;
Exit when c1%NOTFOUND;
DBMS_OUTPUT.PUT_LINE(L_empno||'....'||L_ename);
END Loop;
close c1;
END;
/
```

***Procedures:** A procedure is a PL/SQL block which is compiled and permanently stored in the database for repeated execution.

Syntax:

```
create or replace procedure<procedure_Name>
IS
Begin
-----
-----
-----
END;
/
```

Ex:

```
create or replace procedure p1
IS
Begin
DBMS_OUTPUT.PUT_LINE('HELLO');
END;
/
```

***To execute the procedure:**

SQL>EXEC P1

Procedure can have 3 types of parameters

1. Inparameter
2. OutParameter
3. In Out parameter

1. InParameter:

InParameter's are used to accept values from the user.

Ex: create a procedure which accepts two numbers and display the num?

*Create or replace procedure ADD_NUM(A IN number, B IN number)

```
IS
C number;
Begin
C := A + B;
DBMS_OUTPUT.PUT_LINE('The sum is...'||C);
END;
```

/

*create a procedure which accepts employee number and increment salary by 1000?

Create or replace procedure INC_SAL (A IN number)

IS

Begin

Update emp set sal = sal + 1000 where empno = A;

END;

/

*create a procedure which accepts employee number and display salary?

Create or replace procedure display_sal (L_empno IN emp.empno%TYPE)

IS

L_sal emp.sal%TYPE;

Begin

Select sal INTO L_sal from emp where empno = L_empno;

DBMS_OUTPUT.PUT_LINE('The sal is' || L_sal);

END;

/

*create a procedure which accepts deptno and display the name and location?

Create or replace procedure display_details(L_deptno IN dept.deptno%TYPE)

IS

L_Dname dept.Dname%TYPE;

L_loc dept.loc%TYPE;

Begin

Select Dname, loc INTO L_Dname, L_loc from dept where deptno = L_Deptno;

DBMS_OUTPUT.PUT_LINE('L_Dname| '....'|| L_loc);

Exception

WHEN NO_DATA_FOUND THEN

DBMS_OUTPUT.PUT_LINE('IN value NO');

END;

/

Query to see all the list of procedures:

Select object_NAME from user_objects where object_TYPE = 'PROCEDURE';

To see the source code of a procedure:

Select text from user_source where NAME = 'ADD_NUM';

Spool command:

This command is used to extract the character which we can see in SQL*PLUS environment to a text file.

Ex:

```
SQL>spool c:\sample\abc.txt  
SQL>select * from emp;  
SQL>select * from dept;  
SQL>spool off
```

To edit the procedure:

```
SQL>spool c:\sample\xyz.txt  
SQL>select text from user_SOURCE where name = 'p1';  
SQL>spool off
```

Create a procedure which accepts a number and display its square.

Note: if there are any errors in the source code, then procedure will be created with compilation errors.

Show errors command is used to see the compilation errors.

We can execute multiple procedures act once by using a PL/SQL block.

Ex:

```
Begin  
P1;  
ADD_NUM(20,15);  
DISPLAY_SQUARE(1);  
END;  
/
```

We can remove the procedure by using drop command.

Ex: drop procedure p1;

We can call a procedure using another procedure there exist depending between the procedures.

Ex:

```
Create or replace procedure test11  
IS
```

```

Begin
DBMS_OUTPUT.PUT_LINE('This is from test11');
END;
/
Create or replace procedure sample11
IS
Begin
DBMS_OUTPUT.PUT_LINE('This is from sample11');
END;
/

```

Note:

When we drop test11 procedure, sample11 procedure will become invalid.

****Out parameters:** out parameters are used to return the value to the user.

Ex:

Create a procedure which accepts two numbers and return the sum?

```

Create or replace procedure ret_sum
(A IN number,B IN number, C OUT number)
IS
Begin
C := A + B;
END;
/

```

***Steps to invoke procedures which are having out parameters:**

Step 1: create bind variable

SQL> variable N number

Step 2: execute the procedure

SQL> EXEC Ret_sum(10,20,:N)

Step 3: print bind variable

SQL>print N

Ex:

Create a procedure which accepts a number and return its square?

```

Create or replace procedure ret_square(A IN number,B IN number)
IS
Begin
B := A * A;

```

```
END;
/
SQL> variable N number
SQL> EXEC ret_square(7,:N)
SQL> print N
```

IN out parameters: These parameters are used to accept the value as well as to return the value to user.

Ex: create a procedure which accepts a number and return its square?

```
Create or replace procedure ret_square1(A IN out number)
IS
Begin
A := A * A;
END;
/
```

Step 1: SQL> variable m number

Step 2: initialize bind variable

```
Begin
:m = 5;
END;
/
```

Step 3: EXEC ret_square1(:m);

Step 4: print m;

Note:

The scope of the bind variable is with respect to one session.

Once the environment is closed all the bind variables are lost.

***Functions:**

A function is a named PL/SQL block which must and should return a value.

Syntax: create or replace function<FUNCTION_NAME>(VAR1 datatype,
var2 datatype,.....,varn datatype);

Return datatype

IS

Begin

```
END;
```

```
/
```

Ex: Create a function which accepts two numbers and returns the sum?

```
Create or replace function ADD_NUM_FUN(A number, B number)
```

```
Return number
```

```
IS
```

```
C number;
```

```
Begin
```

```
C := A + B;
```

```
Return C;
```

```
END;
```

```
/
```

Output: function created

Calling:

```
Declare
```

```
N number;
```

```
Begin
```

```
N := ADD_NUM_FUN(10,20);
```

```
DBMS_OUTPUT.PUT_LINE(N);
```

```
END;
```

```
/
```

We can invoke the function using 'select' statement.

Ex: select ADD_NUM_FUN(50,10) from dual;

Functions can be invoked from expression.

Ex: select 100 + ADD_NUM_FUN(50,10) from dual;

We can not have DML commands inside a function.

If a function has DML commands, we get the error when we invoke it.

Functions are preferred to perform calculations.

*create a function which accepts salary and returns tax value. Tax value is 10% of salary?

```
Create or replace function CAL_TAX(L_sal emp.sal%TYPE)
```

```
Return number
```

```
IS
```

```
Begin  
L_TAX = L_sal * 10/100;  
Return L_TAX;  
END;  
/  
Ex: select empno, ename, sal, cal_tax(sal) from emp;
```

Differences between procedure and functions:

Procedures

1. Procedures need not return any value can return one or more than one value.
2. DML commands are allowed.
3. Can not be invoked from select statement.

Functions

1. Must and should return only one value.
2. DML commands are not allowed.
3. Can be invoked from select statement and expressions.

***Query to see list of all the functions:**

Select object_Name from user_objects where object_type = 'function';
Procedures and functions are called sub programs.

***Packages:**

A package is collection of logically related sub programs.
Creation of package invokes two steps.
Step 1: creating package specification.
Step 2: creating package body.

Package specification: package specification contains declaration of sub programs.

Package body: package body contains definition of sub programs.

Example of PKS:

```
create or replace package test_PKG1  
IS  
Procedure p1;  
Procedure display_sal(L_empno IN number);  
Procedure display_details(L_deptno IN emp.deptno%TYPE);  
Function cal_tax(L_sal emp.sal%TYPE)  
Return number;
```

```
END test_PKG1;
```

```
/
```

Example of PKB:

```
Create or replace package body test_PKG1
```

```
IS
```

```
Procedure p1
```

```
IS
```

```
Begin
```

```
DBMS_OUTPUT.PUT_LINE('HELLO WORLD');
```

```
END;
```

```
Procedure display_sal(L_empno IN number)
```

```
IS
```

```
L_sal emp.sal%TYPE;
```

```
Begin
```

```
Select sal into L_sal from emp where empno = L_empno;
```

```
DBMS_OUTPUT.PUT_LINE('The sal is...' || L_sal);
```

```
END;
```

```
Procedure display_details(L_deptno IN emp.deptno%TYPE)
```

```
IS
```

```
Cursor c1
```

```
IS select ename from emp where deptno = L_deptno;
```

```
Begin
```

```
For emp_rec IN c1 LOOP
```

```
DBMS_OUTPUT.PUT_LINE(emp_rec.ename);
```

```
END LOOP;
```

```
END;
```

```
Function cal_tax(L_sal emp.sal%TYPE)
```

```
Return number
```

```
IS
```

```
Begin
```

```
Return(L_sal*10/100);
```

```
END;
```

```
END test_PKG1;
```

```
/
```

To invoke a sub program which is inside the package, we need to mention package

name subprogram name.

Ex: Exec test_PKG1.p1

Select test_PKG1.cal_tax(2500) from dual;

Note:

Packages can not be parameterized.

Packages can not be nested.

Packages help in modularization.

Packages help in overloading subprograms.

Example of overloading sub programs:

Create or replace package test_pkg2

IS

Procedure p1(A number);

Procedure p1(A number, B number);

END test_pkg2;

/

PKB:

Create or replace package body test_pkg2

IS

Procedure p1(A number)

IS

B number;

Begin

B := A * A;

DBMS_OUTPUT.PUT_LINE('The square is ... ' || B);

END;

Procedure p1(A number, B number)

IS

C number;

Begin

C := A + B;

DBMS_OUTPUT.PUT_LINE('The sum is ... ' || C);

END;

END test_pkg2;

/

Invoke the package

```
Begin  
Test_pkg2.p1(10);  
Test_pkg2.p1(10,20);  
END;  
/
```

Dropping the package involves two steps:

Step 1: drop package body
Step 2: drop package specification
Ex: drop package body test_pkg2;
 Package body dropped
Ex: drop package test_pkg2;
 Package dropped.

*****Triggers:**

A Trigger is a PL/SQL block which is executed automatically basing on an event.

Triggering events are inserted, update, delete.

Trigger timing can be before, after, instead of

Syntax:

Create or replace trigger <TRIGGER_NAME><TIMMING><EVENT> ON <OBJECT_NAME>

```
Begin  
-----  
-----  
-----  
END;  
/
```

Ex:

```
create or replace trigger trg1  
after insert on dept  
begin  
DBMS_OUTPUT.PUT_LINE('hello');  
END;  
/  
Trigger created  
SQL> insert into dept values(65,'Admin','HYD');
```

Trigger can be created on multiple events

Ex:

```
Create or replace trigger trg1  
After insert or update or delete on dept  
Begin  
If inserting then  
DBMS_OUTPUT.PUT_LINE('thank you for inserting');  
Elself updating then  
DBMS_OUTPUT.PUT_LINE('thank you for updating');  
Else  
DBMS_OUTPUT.PUT_LINE('thank you for deleting');  
End If;  
END;  
/
```

SQL> insert into dept values(65,'arun','hyd');

Thank you for inserting

SQL> deleting from dept where deptno = 65;

Thank you for deleting

*Triggers are divided into two types

1. Statement level trigger
2. Row level trigger

***Statement level trigger:**

These triggers are executed only once irrespective of number of rows affected by the event. By default every trigger is a statement level.

***Row level trigger:**

These triggers are executed for every row which is affected by the event (multiple numbers of times). We can create a row level trigger by using “FOR EACH ROW” clause.

Ex:

```
Create or replace trigger trg2  
After update on emp for each row  
Begin
```

```
DBMS_OUTPUT.PUT_LINE('thank you for updating');
END;
/
```

Trigger updated

```
SQL> update emp set sal = 2000 where deptno = 30;
```

Output:

```
Thank you for updating
6 rows updated
```

Triggers are used to enforce business rules.

We can enforce business rules by using :OLD and :NEW qualifiers.

***Query to see list of all the triggers:**

```
Select object_Name from user_objects where object_TYPE = 'Trigger';
```

OR

```
Select Trigger_Name from user_Trigger;
```

***To see the source code of a trigger:**

```
Select text from user_source where name = 'trg2';
```

***To drop a trigger:**

Syntax: drop trigger <trigger_name>.

Ex: drop trigger trg2;

***Using: new qualifier:**

*Create a trigger which will allow insert command only when salary is less than 5000.
Trigger should reject the insert command by providing a meaningful message. If SAL > 5000?

Create or replace trigger TRG5 before insert on emp for each row

Begin

If :new.sal>5000 then

```
Raise_application_error(-20150,'sal cannot be more than 5000');
```

END IF;

```
END;
```

```
/
```

Ex: insert into emp(empno, ename, sal, deptno) values(444,'BBB',7500,10);

Output: ERROR ORA-20150 :sal cannot be more than 5000

```
Insert into emp(empno, ename, sal, deptno) values(111,'AAA',2500,10);
```

Output: 1 row created

*create a trigger which will accept insert command only for deptno 10?

```
Create or replace trigger trg3 before insert on emp for each row
```

```
Begin
```

```
IF :new.deptno<>10 then
```

```
Raise_application_error(-20150,'deptno should be only 10');
```

```
END IF;
```

```
END;
```

```
/
```

***Using :old qualifier:**

*A trigger should restrict delete operation on president?

```
Create or replace trigger trg7 before delete on emp for each row
```

```
Begin
```

```
IF :old.JOB = 'president' then
```

```
Raise_Application_Error(-20150,'cannot delete president');
```

```
END IF;
```

```
END;
```

```
/
```

```
Delete from emp where empno = 7839;
```

Error: ORA -20152, cannot delete president

```
Delete from emp where empno = 7499; --- valid
```

*Create a trigger which will restrict don't operations on weekends?

```
Create or replace trigger trg8 before insert or update or delete on emp
```

```
Begin
```

```
IF TO_CHAR(sysdate,'DY') IN ('sat','sun') then
```

```
Raise_Application_Error(-20160,'cannot perform dml operations on weekends');
```

```
END IF;  
END;  
/
```

*Instead of triggers:

Instead of triggers are created on complex view.

By using instead of trigger we can execute insert command on a complex view

Ex:

COMPLEX VIEW

Create view vv1

```
As select e.empno, e.ename, e.sal, e.deptno, d.deptno, d.loc from emp e1 dept d  
where e.deptno = d.deptno;
```

View created

```
Select * from vv1;
```

*Instead of trigger example:

Create or replace trigger trg9 instead of insert on vv1 for each row

Begin

```
Insert into dept values (:new.deptno, :new.dname, :new.loc);
```

```
Insert into emp(empno, ename, sal, deptno) values (:new.empno, :new.ename,  
:new.sal, :new.deptno);
```

END;

```
/
```

Ex: insert into vv1 values (555,'ddd',2000,60,'ADMIN','HYD'); --- valid

Note:

:new and :old qualifiers can be used only in the row level trigger.

*Abstract datatypes:

Abstract data types are consists of one or more subtypes. Rather than being constrained to the standard oracle data types of number, date and varchar2 data types can more accurately describe your data.

Ex:

```
SQL>create type address_ty5 as object  
      (street varchar(20),  
       city varchar2(10),
```

```
        state char(10),
        pin number);
    /
```

Type created

```
SQL> create type person_ty5 as object
        (name varchar2(20),
         Address address_ty5);
    /
```

Type created

```
SQL> create table customer5
        (customer_ID number(3),
         Person person_ty5);
```

```
SQL> insert into customer5 values
        (1,person_ty5('hari',address_ty5('#102 lokhanadala','mumbai','MH',10101)));
```

```
SQL> select customer_ID, c.person.name from customer c;
```

*Nested table:

Nested table is a collection of rows, represented as a column with in the main table. For each record with in the main table, the nested table may contain multiple rows. In one sense, it's a way of storing a one-to-many relationship with in one table.

```
SQL> create or replace type emp_ty5 as object
        (desg varchar2(20),
         dname varchar2(20),
         doj date);
    /
```

Type created

```
SQL> create type emp_nt5 as table of emp_ty5;
    /
```

Table created

```
SQL> create table emp_data5
        (ename varchar2(20),
```

```
details emp_nt5)
nested table details store as emp_nt_tab5);
```

Table created

```
SQL> set describe depth2
```

```
SQL> desc empdata5
```

<u>Name</u>	<u>Type</u>
Ename	Varchar2(20);
Details	Emp_nt
Desg	Varchar2(20)
Dname	Varchar2(20)
Doj	Date

```
SQL> insert into empdata5 values
```

```
(‘Raju’, emp_nt5(‘clerk’,’sales’,’12-sep-05’),
emp_ty5(‘Asst’,’mrket’,’15-oct-04’),
emp_ty5(‘mngr’,’sales’,’13-aug-09’)));
```

```
SQL> select * from emp data5;
```

***VARRAYS:** varrays can also be used to create one-to-many relationship with in the table.

***creating varray:**

Ex: create type dependent_birthday_t5 as varray(10) of date;

Using varray in table:

```
Create table employee5( id number, name varchar2(20),
dependent ages dependent-birthdate-t5);
```

inserting row into table:

```
insert into employee5 values(42,’Arun’, dependent_birthday_t5
(‘12-jan-1765’,’04-jul-1977’,’11-mar-2021’);
```

*****Differences between nested tables and varrays*****

Nested tables

1. There is no restriction on size.
 2. Data is stored in special auxiliary tables called as store tables.
1. We need to define the maximum size.
 2. Data is stored inline to the rest of the table data.

Varrays

***Execute immediate:**

One can call DDL statement like create, drop, truncate and etc from PL/SQL by using the “Execute immediate” statement.

Ex:

```
Begin
  Execute immediate 'drop table dept';
END;
/
```

```
Begin
  Execute immediate 'Truncate table emp';
END;
/
```

***BULK COLLECT:**

Bulk collect feature helps in improving the performance of explicit cursor programs. Fetch statement can fetch all the rows from the cursor to the programs local variable at once thus helps in improving the performance.

Ex:

```
Declare
  Type string_array is varray(20) of varchar2(20);
  L_ename string_array;
  Cursor c1
    Is select ename from emp;
  Begin
    Open c1;
    Fetch c1 bulk collect into L_ename;
    Close c1;
    For i in L_ename.first .. L_ename.last loop
      DBMS_OUTPUT.PUT_LINE(L_ename(i));
    End loop;
  End;
```

/

***REF CURSOR:**

A ref cursor is basically a data type.

A variable created based on such a data type is generally called a cursor variable.

A ref cursor can be associated with more than one select statement at run time.

Before associating a new select statement. we need to close the cursor.

Ex:

Declare

Type r_cursor is REF cursor;

C_emp r_cursor;

Type rec_emp is record{

name varchar2(20);

sal number(6);

};

er rec_emp;

begin

open c_emp for select ename, sal from emp where deptno = 10;

DBMS_OUTPUT.PUT_LINE('department: 10');

DBMS_OUTPUT.PUT_LINE('.....');

Loop

Fetch c_emp into er;

Exit when c_emp%notfound;

DBMS_OUTPUT.PUT_LINE(er.name||'....'||er.sal);

End loop;

Close c_emp;

Open c_emp for select ename, sal from emp where deptno = 20;

DBMS_OUTPUT.PUT_LINE('department: 20');

DBMS_OUTPUT.PUT_LINE('.....');

Loop

Fetch c_emp into er;

Exit when c_emp%notfound;

DBMS_OUTPUT.PUT_LINE(er.name||'-'||er.sal);

End loop;

Close c_emp;

END;

/

SQL * loader:

SQL loader is a tool which is used to load the data from the file to the table.

This tool requires control file(.Ctrl).

Control file contains all the information about source and destination.

It is developer responsibility to create a control file.

Syntax to create control file:

LOAD data

Infile '<Data filepath>'

Insert into table <Table_name> fields Terminated by ','

(col1, col2,...., coln)

Ex:

```
LOAD data
```

```
INFILE 'E:\sunil\student_data.txt'
```

```
Insert into table student50 fields terminated by ','
```

```
(sno, sname, marks)
```

Steps to invoke the tool:

Step 1: open the command prompt.

Step 2: >SQLLDR SCOTT/TIGER

```
CONTROL: E:\SUNIL\Load.CTL
```

***Autonomous transactions:**

In general a commit command used in a PL/SQL block will act globally and make all the changes permanent.

To restrict commit command to a specific program we need to make the PL/SQL block autonomous.

We can create autonomous PL/SQL block by using

'PRAGMA AUTONOMOUS_TRANSACTION' in the declare section.

PRAGMA autonomous_transaction is a compiler directive.

Ex:

```
Create table student20(sno number(3), sname varchar2(10), marks number(3));
```

```
Insert into student20 values(101,'Arun',40)
```

```
Insert into student20 values(102,'Arun',40)
```

```
Declare
```

```
Pragma autonomous_transactions;
```

```
Begin
```

```
Insert into student20 values(103,'Arun',40);
Insert into student20 values(104,'Arun',40);
commit;
END;
/
```

Where current of:

Where current of clause is used in some update statements. The where current of clause is an update or delete statement states that the most recent row fetched from the table should be updated.

We must declare the cursor with 'for update' clause to use this feature.

Ex:

```
Declare
Cursor c1
IS
Select empno, ename, sal from emp
where comm is null
for update of comm.;
var_comm number(4);
begin
for cur_rec.sal < 2000 then
var_comm := 200;
elsif cur_rec.sal < 4000 then
var_comm := 400;
else
var_comm := 100;
end if;
update emp set comm = var_comm
where current of c1;
end loop;
end;
/
```

***Managing large objects:**

***Creating table with LDB columns:**

Ex: create table airbus_desc5(airbusno char(5), airbus_det bfile, airbus_profile clob);

***Insert values in lobs:** To insert values in the bfile, the function bfilename is used. It takes the os path of the directory and the name of the file.

Ex:

Insert into airbus_desc5 values('ABO1', bfilename('E:\sunil','esert.jpg'), 'the description the plane is as follows');

***displaying data from lobs:** data from lobs cannot be displayed, except for clob by using select statement.

Ex: select aitbusno, airbus_profile from airbus_desc5;

***Locks:**

As oracle is a multiuser environment there is always a chance that multiple users will perform DML operators on some table parallel in such case the data becomes inconsistent. To maintain the consistency of data base a user can lock the table.

Select for update command is used for locking a table.

Ex:

Ajit> select * from student for update;

Table is locked by Ajit

ASHWIN> update Ajit.student set marks = 95 where sno = 101;

Client will be in waiting state.

Locks are released when a user executes commit command.

***Levels of locks:** There are three levels of locks.

1. Row level
2. Page level
3. Table level

***Row level:** when where clause in select for update command is evaluating to one row, row level lock is applied.

Ex: select * from student where sno =101 for update;

***Page level:** when where clause in select for update command is evaluating to multiple rows, page level lock is applied.

Ex: Select * from emp where deptno = 20 for update;

***Table level:**

When where clause is not used in select for update, table level lock is applied.

Ex: select * from emp for update;

***FLASHBACK and PURGE command:**

*What is Recycle bin?

Oracle has introduced “Recycle bin” feature oracle log to store all the dropped objects. A user drops a very important table accidentally. Oracle log Recycle bin feature the user can easily restore the dropped object.

***To enable the recycle bin:**

SQL> Alter system set recycle bin = on;

or

SQL> Alter session set recycle bin = on;

To view the recycle bin use the follows command:

SQL> show recycle bin;

Ex:

Create table test_inet1(val number(2));

SQL> insert into test_inet1(val) values(10);

SQL>drop table test_inet1;

Print the recycle bin;

Restore the objects back to data base:

FLASHBACK table <<table_name>> to before drop;

Ex: Flashback tabe test_inet1 to before drop;

SQL> select * from test_RBIN;

***Clearing the recycle bin (RB):**

To clear the recycle bin the following statement can be used.

SQL> purge table <<table_name>>;

SQL> purge table student6;

*****Hibernate*****

Hibernate is a frame work. Hibernate frame work is used to develop a java application to interact with database server.

A frame work is a piece of software this piece of software contains solutions from commonly repeatedly occurred problems occurs multiple projects.

IDE (Integrate Development Environment):

As part of IDE all the components all integrated.

Ex: Editors, Compilers, Servers, Browser etc.

By using IDE we can develop project quickly.

IDE is will improve the productivity of the developer.

The following are the most popular IDE,s.

1. Eclipse
2. My Eclipse
3. Net Beans
4. RAD (Relational Application Development)
5. J Builder

To work with IDE we required a “workspace” folder.

A work space folder contains all the files which are related to that project.

When we double click on Eclipse.exe it will launch a dialog whose name is work space launch. To this work space launch we have to supply work space folder as input.

When we start the Eclipse IDE in the work space folder it has created a folder with the name.metadata. This folder contains all the files which are used by Eclipse IDE.

Eclipse IDE contains set of perspective by default Eclipse IDE launch's JEE perspective.

A perspective contains set of views. A view is a Email window. A perspective contains set of views.

If we would like to perform any operations in IDE we must create a project.

Procedure to create the project in IDE:

Step 1: file new project.

Step 2: select the appropriate project and click on next button.

Step 3: in new java project window enter the project name and click on finish.

Step 4: we delete the project with out checking the check box. The files will not delete permanently.

As part of java prospective we use out line view to see all the variables and methods which are available in class.

The package explorer view is used Ctrl + O files which are available in the project.

Select → Project → Properties → java Build path → Library → add external jar files.

In IDE we can disable auto build process (Project → builds automatically).

Note:

We can change the short cuts of Eclipse IDE
(Window → Preferences → General → Keys).

As part of a Eclipse IDE we can replace or get the old code by using compare with or replace with local history.

To create a jar files we use an option export (files → export).

***Procedure to create web based application:**

Step 1: file → new → web project.

Step 2: The above step will display a dialog with the name ‘new web project’. In that dialog enter the project name make sure that project name and web root folder name is same.

When we work with IDE to configure in deployment descriptor we have two options. They are like source and design. To see the contents of web.xml file we can use an option source.

By using graphical user interface if we want configure servlets. We can use design view. When we use IDE we no need to deploy the project manually. We can configure IDE to deploy the project. As part of JEE respective. we can use server view to configure the project in IDE.

Step 1: Get the server view and right click chooses an option configure server connector.

Step 2: Choose the appropriate server the home directory of the server. Same servers required user name and password supply these values and click on “OK”.

Step 3: To add the project to server we can use an option add deployment.

Step 4: There are some IDE's are available which are responsible to interact with database

Servers. The popular IDE's for oracle database server are TOAD and SQL developer.

For the database server like MYSQL we can use “MYSQL workbench”.

Step 5: In the companies because of licenses issues we may not use the IDE like TOAD.

Eclipse gives has provided a feature to interact with database servers. We can use Eclipse/MYECLIPSE to interact with database servers.

***Procedures to configure IDE to interact with database server:**

1. Open MYECLIPSE database explorer of prospective.

2. In the database browser view when we right click it launches a popup menu from that choose an option new.
3. The above step as lunched a window whose name is database driver from that choose driver template and provide a driver name. Now we have to supply driver class, url, username, password to interact with DB server.

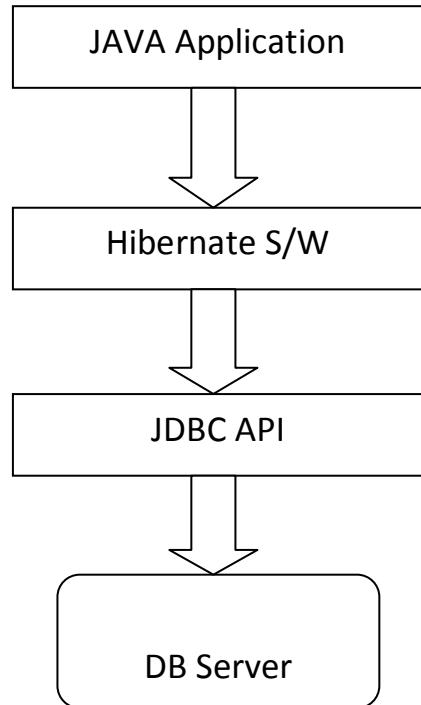
***Hibernate is a frame which is used to interact with database server:**

There are so many frameworks are available in the market there are some of struts, springs, JSF etc.

1. Struts, spring frame works are used to develop web-based applications.
2. Hibernate is a technology are a framework which is used to interact with database server this framework resolves all the problems of JDBC. In a project if he tries to use hibernate frame work if he develop the project quickly.
3. As part of JDBC we have to write lot of code to hardly checked Exception.
4. As part of Hibernate we no need to provide the code to handle checked Exception.
5. If we use JDBC a programmer is responsible to register the driver and get the connection. Programmer must provide the code to close the connection. As part of Hibernate he is responsible to open the connection and close the connection.
6. As part of Hibernate it's only starts the transaction and ends the transaction. Hibernate internally uses JTA (Java Transaction API). To start the transaction and end the transaction.
7. Hibernate support internal connection provides Hibernate uses C3P0 connection pool. DBCP connection pool. If we want to use external connection pool like web-logical we can configure it. If we use JDBC we have to provide the code to handle database specific error code.
8. If we use Hibernate we no need to provide the code to handle database specific errors.
9. Hibernate supports its own query language HQL (Hibernate Query language) to resolve the problems of some java application to interact with multiple DB Servers without changing the queries.
10. In case of JDBC we have to provide the code to remove the hard coding. In case of Hibernate we provide the details in Hibernate configuration by reading the details from configuration file (.XML). It will interact with multiple Database Server.

11. If we use JDBC programmer is responsible to write the code to generate the primary key values. As part of Hibernate they have provided pre-defined class to generate primary key values.
12. By using Hibernate we can represent the queries in the form of object or we can use criteria API. As part of Hibernate we can achieve polymorphism between the tables and we can achieve association also.
13. *by using JDBC we can't transfer result set object from one program to another program (result set can't be transferable) to resolve this problem we have to provide the code to retrieve data from result set object and store it in array list object if we use Hibernate. Hibernate does all these work.
14. *we have the technology like EJB to interact with DB Server we can't run EJB Application without EJB container. We can't use EJB's in all type of application we can resolve all these problems we can use Hibernate.

*The following is true Architecture of Hibernate framework.



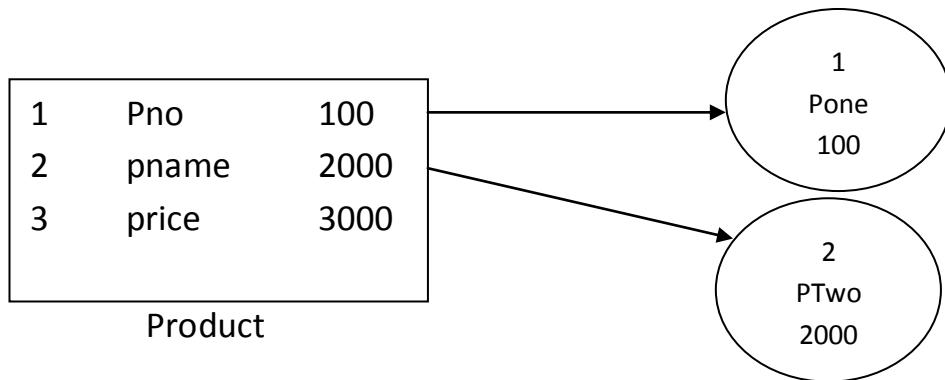
*Hibernate is called as ORM tool are ORM framework ORM (Object Relational Mapping).

In the market they are so many ORM frame works are available some of them are Hibernate, JPA (java persistent API) JDO (java data object) i batches, STO (service data o/p), Top Link etc.

All these frame works are used as ORM Tools. The differences between the Tools are the names of the classes are different.

In the database server the data will be stored in the side the tables between the tables we establish a relationship because of this reason we call it is relational data.

*Representations relational records in the form of object we call it has ORM:



*can we use JDBC to represent records in the form of object?

- A. Yes, we can use JDBC to represent the records in the form of object. In case of JDBC we have to provide huge amount of code to represent huge amount of data in the form of object.
- B. As a Hibernate developer we need to develop a java application with uses Hibernate API and interact with database server. If we want to develop a java application we need to configure DB server and provide some additional files to Hibernate software.
- C. If the Hibernate software to communicate with database server we required a user with special privileged (sequence) the tables in the database server must contain primary keys.

***Procedure to create user and assign some privileges:**

1. Login to database server using administrative user use the following command assign the user and privileges.

```
SQL> create user hib identified by abc;  
Grant connect, resource to hib;  
Grant create sequence to hib;
```

2. We have to create the table with primary key.

```
SQL> create table emp (eno number(4) primary key, ename varchar2(20),  
Address varchar2(20));
```

```
SQL> create table product (pid number(5), name varchar2(20), price number(10,2));
```

```
SQL> alter table product add primary key(pid);
```

3. We can get Hibernate software from Hibernate.org
4. Hibernate software is a collection of jar files the most important jar file is Hibernate3.jar
5. If we want to use Hibernate software we have to develop the following 3 files.
 - 5.1. Hibernate configuration file
 - 5.2. Java Beans (POJO classes) (plain Old Java Object)
 - 5.3. Hibernate Mapping files (hbm)
6. Every framework uses configure file. This is the starting point to any framework generally the configuration files or xml files(we can use property files also as configuration files)
7. *in Hibernate we use Hibernate.cfg.xml as the configuration file we can change the configuration file name according to requirement.
8. Hibernate configure file contains the entire information which is required to communicate with database server. They are:
 - a. Driver class
 - b. url
 - c. username
 - d. pwd
 - e. dialect

```
driver_class =oracle.jdbc  
url = jdbc.ora....  
username = hib  
password = abc  
dialect = OracleDialect  
//information about HRM files
```

Hibernate.cfg.xml

9. Hibernate software required the java program to represent a record in the form of object. Because of this reason we have to develop couple of java programs this is based on the number of tables available in the database server.
10. These programs contains instance variables, setRow() and gerRow() methods. Developing these programs are easy because of this reason POJO classes are known as (plain old java objects).

11.In our database server we have two tables emp and product. We need to develop two pojo classes.

- a. Emp.java
- b. Product.java

Note: There is no rule saying protocol names and tables name must be same.

```
public class employee{  
    int eno;  
    string ename;  
    string eddress;  
    public void set eno(int eno){  
        this.eno = eno;  
    }  
    public int geteno(){  
        return eno;  
    }  
}
```

12.We have to develop Hibernate mapping files. These files contain. The information about which protocol class is mapped with which table and which properties mapped with which column.

Employee (POJO)	→	Emp (Table)
EMPNO	→	ENO
EMPNAME	→	NAME
EMPADDRESS	→	ADDRESS

emp.hbm.xml

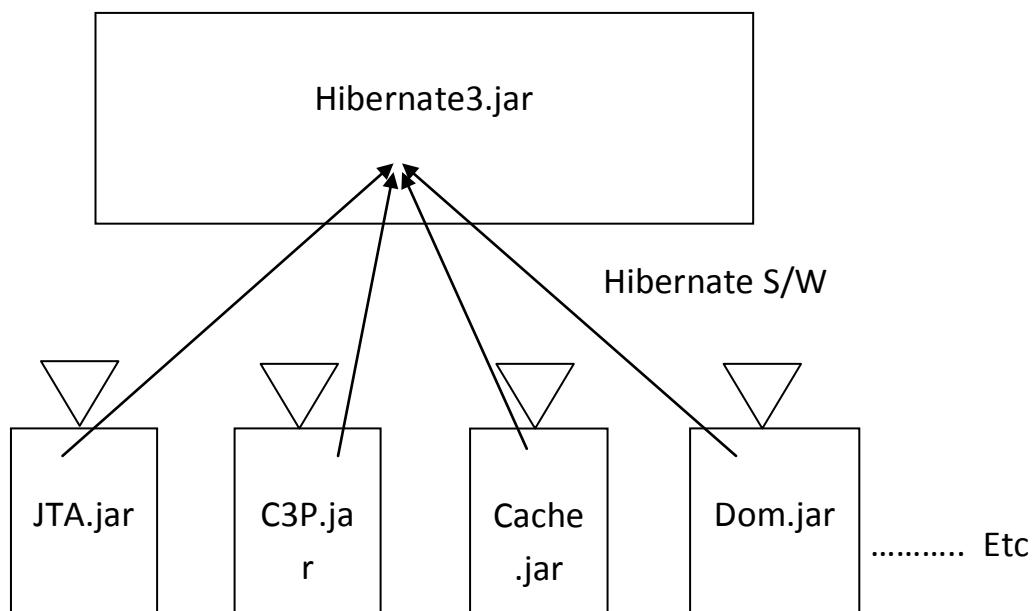
Note:

IDE takes care of generating Hibernate configuration file, pojo classes and Hibernate Mapping files.

***The POJO classes must following rules:**

1. Must contain a default constructor.
2. Must be placed inside a package.

3. The POJO class supports properties. A setRow() and getRow() methods are called as properties.
4. When the Hibernate people are developing software they are started using the help of other software's. They are like Transaction API, Connection Tool, Dom Parse, cache jar files. If we would like to run the Hibernate software are need to make sure that all these jar files generate the CLASSPATH otherwise Hibernate software will fail to run.



***Procedure to configure the Hibernate software and generate Hibernate configuration file by using MyEclipse IDE.

- I. Start MyEclipse IDE pointing to works place folder.
- II. Configure MyEclipse IDE to interact with the DB Server.(create DB Browser)
- III. Create a java project and add the package to it.
- IV. Add Hibernate capabilities to the above project. MyEclipse → project capabilities
→ Hibernate capabilities.
- V. Choose the required jar files to be added. (Core library and notations).
- VI. Choose new Hibernate configuration option and click on next button. From the list of available DB Driver select to which database server we have to connect.
- VII. Choose the package name and provide the class name as ORASF and click on finish button.

***Procedure to generate hbm files and POJO classes:**

Step 1: Go to MyEclipse db explorer perspective and establish the connection with Database server.

Step 2: Select the all required tables and right click on it generates @ launch pop-up menu from that choose on option Hibernate reverse Engineering.

Step 3: Select src and package and check the first 3 check boxes and click on finish button launches abstract class.

Step 4: The following is the configuration is Hibernate configuration file.

```
<hibernate-Configuration>
    <Session-Factory>
        <property name = "Connection.driver_class"> oracle.jdbc.driver.OracleDriver
        </property>
        <property name = "Connnection.url"> jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name = "Connection.username">hib</property>
        <property name = "dialect">org.hibernate.dialect Oracle9iDialect</property>
        <mapping resource = "info/inetsolv/product.hbm.xml"/>
        <map resource = "info/inetSolv/Emp.hbm.xml"/>
    </Session-Factory>
</hibernate-configuration>      // Save hibernate.cfg.xml
```

Step 5: The following is the POJO class of emp tables.

```
public class Emp{
    private Integer eno;
    private String name;
    private Double salary;
    // provide setters and getters methods
    // above instance variables.
}
```

// Save Emp.java

Step 6: The following on the tags of Employee file.

```
<hibernate mapping>
    <class name = "info.inetsolv.Emp" table = "Emp">
        <id name = "eno" type = "java.lang.Integer">
```

```
<column.name = "ENO" presission = "s"/>
<generator class = "assigned"/>
</id>
<property name = "name" type = "java.lang.String">
    <column name = "NAME"/>
</property>
<property name = "salary" type = "java.lang.Double">
    <column name = "SALARY"/>
</property>
```

*The following are the most important interfaces and classes.

***Interfaces:**

1. org.hibernate.SessionFactory
2. org.hibernate.Session
3. org.hibernate.Transaction
4. org.hibernate.Query
5. org.hibernate.Criteria

***Classes:**

1. org.hibernate.cfg.Configuration

*The following are the steps which will be used to develop Hibernate application.

- 1) Create Configuration Object.
- 2) Call the Configuration() method by using the Configuration Object.
- 3) Get SessionFactory Object by using Configuration Object, we use a method buildSessionFactory().
- 4) Get Session Object by using SessionFactory. Call the method openSession().
- 5) Get the Transaction Object.
- 6) Create the POJO class Object which we would like to perform the Operations.
- 7) Store the data in the POJO class Object.
- 8) Call the methods save/update/delete/load methods.
- 9) End Transaction by using commit/rollback.
- 10) Close Session Object.
- 11) Close SessionFactory Object.

***Requirement:**

Develop a Hibernate application to insert records in to the emp table. The following is the java code which is used to store the data into emp table.

```
public class StoreEmpDetails{  
    public static void main(String args[]){  
        Configuration cfg = new Configuration();  
        cfg.configure();  
        SessionFactory sf = cfg.buildSessionFactory();  
        Session hsession = sf.openSession();  
        Transaction tx = hsession.beginTransaction();  
        Emp e = new Emp();  
        e.setEno(1);  
        e.setName("Raju");  
        e.setAddress("Hyd");  
        hsession.save(e);  
        tx.commit();  
        hsession.close();  
        sf.close();  
    }  
}
```

Note:

When we Run the same application for two times we got an Exception saying that ConstraintViolationException.

1. When we create the configuration object we are create an environment to store the configuration details. In this configuration object we can store Driver class, url, username, password and mapping information.
2. When we call cfg.configure it checks for hibernate.cfg.xml file in the CLASSPATH. If it is available it start reading the contains from hibernate configuration file. Now the hibernate file the corresponding hbm files it opens all the hbm files and try to read the contents from all hibernate mapping files. All this information stored a JVM's memory (configuration object). If the configuration object not available in the class objects it throw Exception org.hibernate.HibernateException.

Note:

This method is an Expensive operation. In a project it is recommended to call the configure method only once in the project life cycle.

```
public class StoreEmpDetails{  
    public static void main(String args[]){  
        Configuration cfg = new Configuration();  
        cfg.configure();  
        SessionFactory sf = cfg.buildSessionFactory();  
        Session hsession = sf.openSession();  
        Transaction tx = hsession.beginTransaction();  
        Emp e = new Emp();  
        Product p = new Product();  
        e.setEno(1);  
        e.setName("Raju");  
        e.setAddress("Hyd");  
        p.setId(1);  
        p.setName("Rice");  
        p.setAmount(1000);  
        hsession.save(e);  
        hsession.save(p);  
        tx.commit();  
        hsession.close();  
        sf.close();  
    }  
}
```

This method is an expensive operation in a project it is recommended to call the configure method only once in the project life cycle.

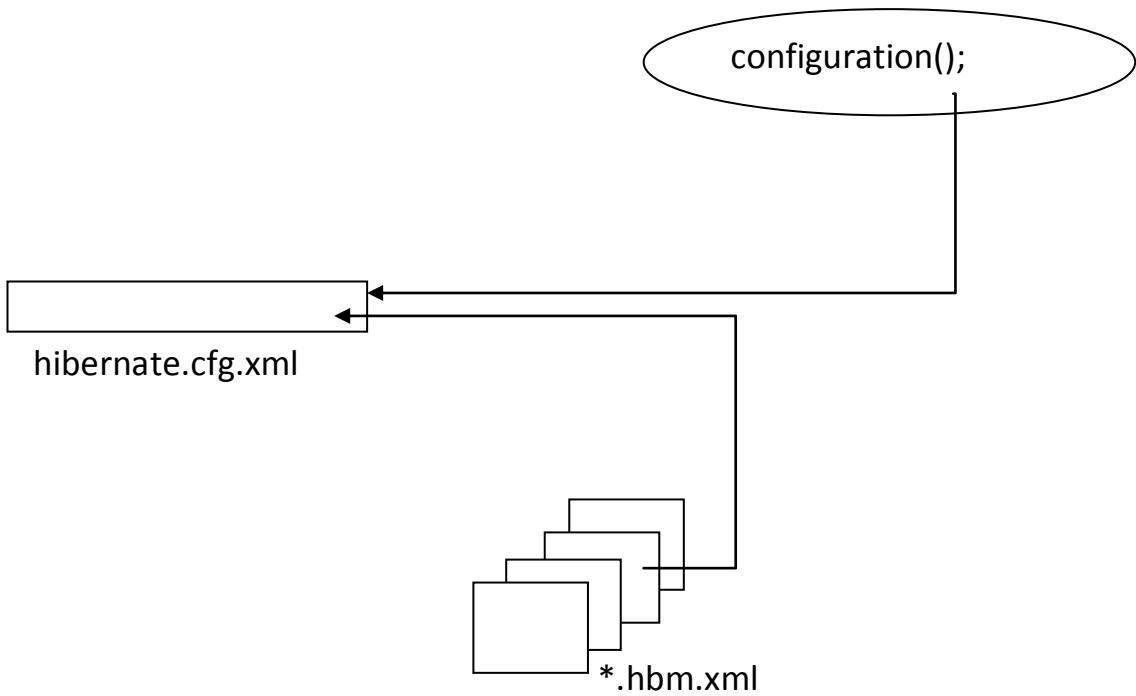
In any frame wrote we can change the configuration file name according to our requirement.

If you are using our own configuration file name we have to use the over loaded configuration method.

Ex: cfg.configure("myproject.xml");

Note:

The default configure method always check for hibernate cfg.xml file. It always recommends using .cfg in the configuration file name. Because of this we can easily recognize the configuration file name.



When we call `cfg.buildSessionFactory()` method it gets driver class, url, username and password these values are supplied as input to hibernate internal connection pool. Now the hibernate internal connection pool. Get the connections from database servers. Now the `buildSessionFactory` method gets a connection from connection pool and establish the connection with database server. It will check whether all the required tables are available or not. If not available if required `buildSessionFactory()` method create the tables. It is the responsibility of hibernate `buildSessionFactory()` to create all the 'CURD' queries for every table and store the queries in JVM's memory now the `buildSessionFactory` close the connection.

Note: calling `buildSessionFactory()` method repeatedly in the project is not recommended. It is recommended to call only once in life time of the project.

By default `buildSessionFactory()` method is not creating the tables. If you want to hibernate to create the tables we have to supply an additional property '`hbm2ddl.auto`' these properties can take any of the following four values create, update and create-delete, validate.

The following is the tag which has to be added to Hibernate configuration file.

```
<property name = "hbm2ddl.auto">update</property>
```

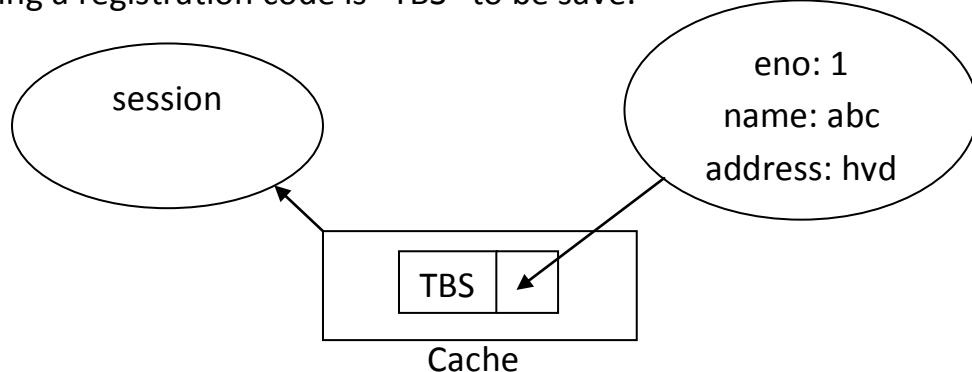
It is always advisable to get a session Object whenever we would like to carry out any work by using Hibernate. It's always recommended to close the session Object after we

finish the work. Getting a session object is similar to getting a connection object in JDBC.

Whenever the session object is created immediately Hibernate starts a cache object and it will be associated to session object are all this cache objects as 1st level cache. Hibernate removes the cache object when ever we close the session object.

Arrays we have to start transaction after the session object is created. We need to start the transaction only for insert/update/delete operation only. We no need to start the transactions for retrieve the records.

When we call hsession save hibernate takes the object and add it to first level cache by using a registration code is “TBS” to be save.



When we call “tx.commit” hibernate got first level cache and check are there any object are available in first level cache if the objects are available hibernate check. The registration code of the object hibernate find to which POJO class this object is created and to which table this POJO class is mapped.

Based on the table name and registration and hibernate get the insert query from the JVM's memory. It is the responsibility hibernates to replace positional parameters with appropriate values from the object. Now the hibernate add the query to batch object.

Now the hibernate send the batch object to Database server if it got execute successfully it returns an identifies value. If the batch is failed it throws an exception batch update exception.

Hibernate sends a sql query to Database server to see the sql query sent by hibernate we can add property “show-sql”. This attribute takes a Boolean value.

Ex:

```
<property name = "show_sql">true</property>
```

To see all the messages are work done by hibernate we can use log4j.properties.

```
log4j.root logger = DEBUG/A1  
log4j.appender.A1 = org.APache.log4j.console Appender  
log4j.appender.A1.layout = org.apache.log4j.simple layout
```

We can use a method “persist” to store the data into database server.

Syntax: void persist(object)

Ex: serializable save(object)

When we generate hbm files and POJO classes in hibernate by using IDE based on the column data type IDE generate the appropriate data type in POJO class for example eno number(15) IDE uses big decimal data type for eno number(2) IDE uses byte and etc.

As part of java5.0 sun micro system as added a feature auto boxing. The advantage of auto boxing is we LAN directly convert primitive data type values to wrapper classes.

Ex:

```
int a = 10;  
integer i = a;  
system.out.println(i);
```

When we trying to dual with auto boxing for double data type as shown below we are getting the compilation error.

Ex: Double d = 20d;

*develop a Hibernate application to retrieve a record from emp table where eno is 2?

```
public class RetrieveRecords{  
public static void main(String[] args){  
Configuration cfg = new Configuration();  
cfg.configure();  
SessionFactory sf = cfg.buildSessionFactory();  
Session hsession = sf.openSession();  
Emp e = new Emp();
```

```

Hsession.load(e,new BigDecimal(2));
System.out.println("e.getEno()");
System.out.println("e.getName()");
System.out.println("e.getSalary()");
hession.close();
}
}

```

When we call the load() method the internal code of Hibernate has performed the following steps.

Step 1: It has checked the corresponding POJO class name for the supplied object.
 Step 2: It has checked this POJO class object is mapped to which table for that table
 Hibernate has picked appropriate select query. The following is the query available in
 JVM's memory.

- Select eno, name, salary from emp where eno =? Now the Hibernate has replaced the positional parameter value with 2 and send the query to database server.
- Database server has executed the select query and represented the records in the result set object and given into hibernate software.
- Hibernate software has taken from ResultSet object and by using the getter method got the data from ResultSet object and stored it in POJO class object.
- Now the Hibernate added the POJO class object to 1st level cache.

Note: If we try to call a load cs method on a non available record Hibernate throw an exception saying 'org.hibernate.ObjectNotFoundException'.

To check weather any object is available in 1st level cache or not we can use a method 'contains()'.

Syntax: boolean Contains(object)

*Write a hibernate application to delete the records from product table whose productID is 11. To delete the records from hibernate we can use two approaches.

Approach 1: Load the record and mark the object as to delete.

```

public class DeleteRecords{
public static void main(String[] args){
Configuration cfg = new Configuration();
cfg.configure();
SessionFactory sf = cfg.buildSessionFactory();

```

```

Session hsession = sf.openSession();
Transaction tx = hsession.beginTransaction();
Product p = new Product();
hsession.load(p,11); // step 1
hsession.delete(p); // step 2
tx.commit(); // step 3
hsession.close();
}
}

```

Step 1: when step 1 is executed it has retrieve the records whose primary key value is 11 and add into 1st level cache.

Step 2: When we call the method object is marked as to be deleted.

Step 3: when we call the commit method the hibernate software got the delete query and replaces the positional parameter with primary key value and send the query to database server.

In this approach first we are checked in whether the record is available or not if the record is not available. The load() method throws object not found exception.

Approach 2: Create the POJO class object and supply the primary key value. Mark the POJO class object as to be deleting by calling the delete method.

Ex:

```

public class DeleteRecords1{
public static void main(String[] args){
Configuration cfg = new Configuration();
cfg.configure();
SessionFactory sf = cfg.buildSessionFactory();
Session hsession = sf.openSession();
Product p = new Product();
p.setpid(11);
hsession.delete(p);
tx.commit();
hsession.close();
}
}

```

In this approach when we call the delete method object is marked as to be deleted. When we call the commit() method it has perform the following 3 steps.

Step 1: It check weather primary key value is available in the supplied object or not. If not available it will not carry out any work.

Step 2: If the primary key value is available a select query will be send to database server to check weather record is available or not.

Step 3: If the record is available in DB hibernate send the delete query. If the record is not available hibernate will not do any work.

Approach 1: Updating a record into database server which ever the record we would like to update load the record by calling load() method modify the values by using setter methods in the loaded POJO class object. Now mark the object as to be updated.

Ex:

```
public class UpdateRecord{  
    public static void main(String[] args){  
        Configuration cfg = new Configuration();  
        cfg.configure();  
        SessionFactory sf = cfg.buildSessionFactory();  
        Session hsession = sf.openSession();  
        Emp e = new Emp();  
        hsession.load(e, new BigDecimal(2));  
        e.setName("Raju");  
        hsession.update(e);  
        tx.commit();  
        hsession.close();  
    }  
}
```

Approach 2: Hibernate uses a directory object technique to check weather object value is modified or not. If the value is not modified. Hibernate will not send any update query to the database server. If the values are modified Hibernate send an update query to database server.

Approach 3: In this approach create POJO class object to the class which we would like to update the record and store the data into POJO class object. We need to mark the object as to be updated.

Ex:

```
public class UpdateRecord{  
    public static void main(String[] args){  
        Configuration cfg = new Configuration();  
        cfg.configure();  
        SessionFactory sf = cfg.buildSessionFactory();  
        Session hsession = sf.openSession();  
        Emp e = new Emp();  
        e.setEno(new BigDecimal(20));  
        e.setName("ttt");  
        hsession.update(e);  
        tx.commit();  
        hsession.close();  
    }  
}
```

***evict()**: evict() method is used to remove a specified object from the 1st level cache.

Ex:

```
Transaction tx = hsession.beginTransaction();  
hsession.load(e,new BigDecimal(1));  
e.setName("Raju");  
hsession.evict(e);  
tx.commit();
```

When we run the above application with out evict() method. It has update a record into database server. When we run the some application with evict() method. It has removed employee object from 1st level cache.

***merge()**: merge method is used to add a specified object to the 1st level cache.

Ex:

```
Emp e = new Emp();  
e.setEno(new BigDecimal(22));  
e.setName("ABC modified");  
e.setSalary(1234d);  
hsession.merge(e);  
tx.commit();
```

When the merge() method is called the object is added to 1st level cache without registration code. When tx.commit() method is called it will get the object which does not contain the registration code. It will check whether the object is available in database server by sending select query. If the record is not available it will send an insert query to database server. If the record is already available it will send a update query to database server.

*There three states are available to hibernate objects they are:

1. Transient
2. Persistent
3. Detached

Transient: An object is which is not associated with any session object.

Persistent: An object which is added to 1st level cache is called as persistent state.

Detached: An object which is removed from 1st level cache is called detached state.

***Clear():**

Clear is used to remove all the objects from 1st level cache. This will remove unperformed operations like save and update also. The clear() method will evict all available objects.

***Connection():**

This method is used to get legacy database connection. Generally this is not recommended approach in hibernate. We use this to perform some operations which can not be done by using hibernate.

Ex:

```
Session hsession = sf.openSession();
Transaction tx = hsession.beginTransaction();
Connection con = hsession.Connection();
Statement stmt = con.createStatement();
stmt.executeUpdate("Insert into emp values(2,'sadaf',234)");
tx.commit();
```

In Hibernate when we get the connection object by default auto commit mode to false. We have multiple over loaded methods as for session interface they are:

```
void load(object, pid)
object load(class, serializable)
```

Ex:

```
Class c = class.forName("info.inetsolv.product");
Object o = hsession.load(c,l);
Product p = (product)o;
System.out.println(p.getPid());
System.out.println(p.getName());
System.out.println(p.getPrice());
```

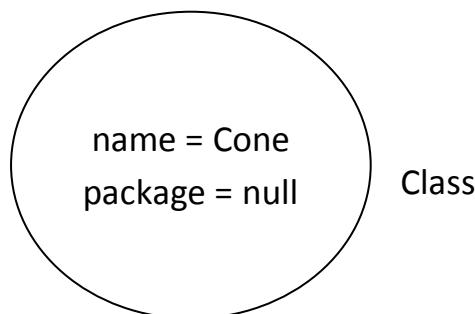
When we call the above load() method if the record is available load() method creates POJO class object and store the data and return POJO class object if the record is not available load() method will not create POJO class object.

We have a static variable class as part of object class whenever we call that variable by using class name it returns the calling class classobject.

Ex:

Cone.class

When the above code is executed it has return class object.



*get():

get() method is also used to retrieve the record from database server if the record is available it returns that POJO class object. If record is not available it returns null value.

Ex:

```
Object o = hsession.get(product.class,2);
If(o!=null){
Product p = (product)o;
System.out.println(p.getPid());
System.out.println(p.getName());
System.out.println(p.getPrice());
}
else{
System.out.println("Record is not available");
}
```

***flush():**

When we call in flush() method all the objects which are available in 1st level cache will be converted into queries and send to database server. Flush will not store data permanently. When we call the commit() method the data is stored permanently.

HbmaddL.auto property takes by different values

1. Update
2. Create
3. Create-drop
4. Validate

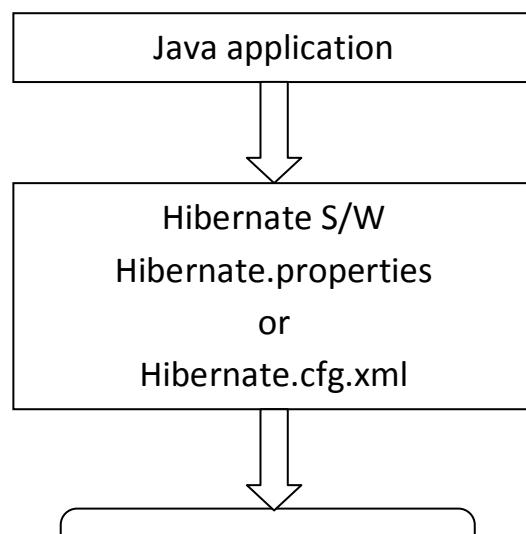
If hbmaddl.auto = update and weather the buildSessionfactory() method is executed it checks weather tables are available or not. If not available it creates the tables.

If hbmaddl.auto = create if the tables are not available buildSessionFactory() method creates it. If the tables are available again if hbmaddl.auto = create-drop if the tables are not available it creates the tables. when we close the session factory object the tables will be dropped.

If hbmaddl.auto = validate, buildSessionFactory() method check weather the tables are present in the database server or not. If not available it will throw an error message missing table.

***Developing Hibernate application by using manual procedure:**

The following is an architecture of hibernate application.



Step 1: Create a table
Step 2: Get the Hibernate software and place in a lib folder (copy ojdbc14.jar also).

Note:

We can get the jar files from IDE.

Step 3: Develop a cmd file which contains the class path to all the Hibernate related jar files.

Ex: Set CLASSPATH=lib\antlr-2.7.6.jar;lib\C3P0-0.9.1.jar;

Step 4: Create the POJO class. We can use any class name as POJO class name for example.

```
public class Employee{  
    int empNo;  
    String empName;  
    double empSalary;  
    public void setEmployeeNo(int employeeNo){  
        this.empNo = employeeNo;  
    }  
    public int getEmployeeNo(){  
        return empno;  
    }  
}
```

*Create hbm file the name of hbm file can be any thing.

```
<?xml version = “1.0”?>  
<!DOCTYPE hibernate.mapping public “something”  
      “http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd”>  
<hibernate-mapping>  
<class name = “info.inetsolv.Employee”table = “emp”>  
<id name = “employeeNo”>  
<column name = “eno”/>  
<generator class = “assigned”/>  
</id>  
<property name = “employeeName”>  
<column name = “name”/>  
</property>  
.....  
</class>  
</hibernate-mapping>
```

*Develop hibernate configuration file.

```
<? xml version = “1.0”?>  
<!DOCTYPE hibernate-configuration public “hibernate-configuration”,  
      “http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd”>
```

```
<hibernate-configuration>
<session factory>
.....
</session factory>
</hibernate-configuration>
```

Develop a java application to store the data into database server.

The parser program check for hibernate dtd file as part of hibernate file.

As part of hbm file we can remove the column tag or attribute if the POJO class properties and column names are same.

*The following is sample configuration for emp table.

```
<hibernate-mapping>
<class name = "info.inetsolv.Employee">
<id name = "employeeNo"/>
<property name = "employeeName"/>
<property name = "employeeSalary"/>
</class>
</hibernate-mapping>
```

We can club multiple hbm files into a simple hbm file. But this approach is not recommended for big projects, it is recommended to use one hbm file BrOne one POJO class.

```
<hibernate-mapping>
<class name = "info.inetsolv.Employee">
<class name = "info.inetsolv.Product">
<id name = "Pid" access = "field"/>
<property name = "name" class = "field"/>
</class>
</hibernate-mapping>
```

We can develop hibernate application with out hibernate configuration file. But we have to provide the properties and mapping files through the java program.

Ex:

```
public class Store{
public static void main(String[] args){
Configuration cfg = new Configuration();
```

```

cfg.setProperty("hibernate.connection.driver_class","oracle.jdbc.driver.OracleDriver");
cfg.setProperty("hibernate.connection.url","jdbc:oracle:thin:@localhost:1521:xe:");
cfg.setProperty("hibernate.connection.username","hib");
cfg.setProperty("hibernate.connection.password","abc");
cfg.setProperty("hibernate.dialect","org.hibernate.dialect.Oracle9Dialect");
cfg.setProperty("hibernate.show_sql","true");
cfg.addResource("a.hbm.xml");
sessionFactory sf = cfg.buildSessionFactory();
session hession = sf.openSession();

.....
.....
...
}

}

```

Instead of addResource() method we can use addClass() method.

Ex:

```
cfg.addClass(info.inetsolv.product.class);
```

when we use addClass() it will check for info.inetsolv.product.hbm.xml file.

The disadvantage of programmatic configuration is when Hard coding the values in the java program. If we want to communicate with same hibernate application with different database server. We have to change the java code because of this reason this approach is not recommended.

Generally in the projects we use properties files to remove hard coding. Most of the projects uses property file end with an extension dot(.) properties inside the properties file we supply a data in the form of key and value.

Ex:

```
Key = value //myproject.properties
```

We can configuration hibernate parameters as part of a property-file. Who's name is hibernate properties.

Ex:

```

hibernate.connection.driver_class = oracle.jdbc.driver.OracleDriver
hibernate.connection.url = jdbc:oracle:thin:@localhost:1521:xe
hibernate.connection.username = hib
hibernate.connection.password = abc
hibernate.dialect = org.hibernate.dialect.Oracle9Dialect

```

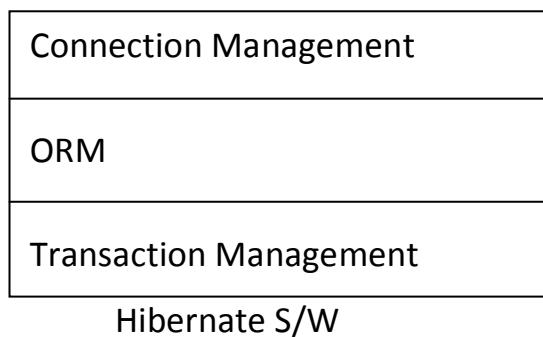
```
hibernate.show_sql = true  
                      //save hibernate.properties
```

It's not recommended to use property file as part of hibernate. This is because as part of the property file are can't configuration the mapping resource files.

We can supply the values by the properties using system properties when we run the application.

Ex: -Dhibernate.connection.driver_class = oracle.jdbc.driver.OracleDriver

Meagerly hibernate is divided into three parts. They are:



Hibernate S/W is good at ORM as well as transaction management the internal hibernate code uses two connections pools C3P, DBCP. It's not recommended to use there connection pools. It's always recommended to use the external connection pool like weblogic connection pool.

*Using procedure to use weblogic connection pool program in hibernate.

1. Configure weblogic server connection pool by specify JNDI name.
2. Get the hibernate S/W place in lib folder and set the class path.
3. Create POJO class and hbm file.
4. Create hibernate configuration file with data source, username and password and jndi.class and jndi.url

```
<hibernate-configuration>  
  <Session-Factory>  
    <property name="hibernate.connection.datasource">mypool</property>  
    <property name="hibernate.connection.username">admin</property>  
    <property name="hibernate.connection.password">inetolv</property>  
    <property name="hibernate.jndi.class">weblogic.jndi.WLInitialContextFactory</property>
```

```
<property name="hibernate.jndi.url">t3://localhost7001/</property>
<property name="hibernate.show_sql">true</property>
<mapping resource = "a.hbm.xml"/>
</Session-Factory>
</hibernate-configuration>
```

5. Develop the hibernate application to store the data into database server.
Note: we have set the class path to web.jar or set DomainEnv.cmd
6. When we are configuration are using connection pool as part of hibernate we have choose use jndi data source rather then jdbc driver.
7. To set the class path to resolve the problem of weblogic.jndi.WLInitialContextFactory we are added weblogic.jar in the class path.
8. Hibernate can be used to communicate with any DB server.

***Procedure to develop hibernate application to interact with MYSQL:**

When we communicate with MYSQL DB Server. We have change the url, username, driver class and etc.

We have to always use "wrapper classes" as part of POJO classes instead of primitive data types. Primitive data types occupy less amount of memory when compared with wrapper classes. Primitive data types can't hold a null value.

Ex: public class MyApp{

```
public static void main(String[] args){
    int a = null;
    System.out.println(a);
}
```

When we compile the above program the compiler reports error message saying null value can not be assigned to primitive data types. we can assigned a null value to wrapper classes.

Ex:

```
public class MyApp{
    public static void main(String[] args){
        Integer a = null;
        System.out.println(a);
    }
}
```

As part of the IDE when we generating hbm files and POJO classes we have an option. We have primitive data types or wrapper classes. They are java types and hibernate types.

When we choose java types it uses wrapper classes. When we choose hibernate types it uses primitive data types.

We are try to use primitive data types and trying to store data into tables. In these scenarios the default values of primitive data types are getting store into data base server.

Ex:

```
Product p = new Product();
p.setPid(Pid);
p.setName(Pname);
hsession.save(p);
```

When we execute the above code even though when the user is not supplied price value application has stored 0.0 values.

When we use primitive data types to retrieve records from a table which contains null values. then the application throwing an exception.

“org.hibernate.propertyAccessException”.

By using JDBC also we can represent the records in the form of objects.

The following is an example of representing data in the form of object by using JDBC?

Ex:

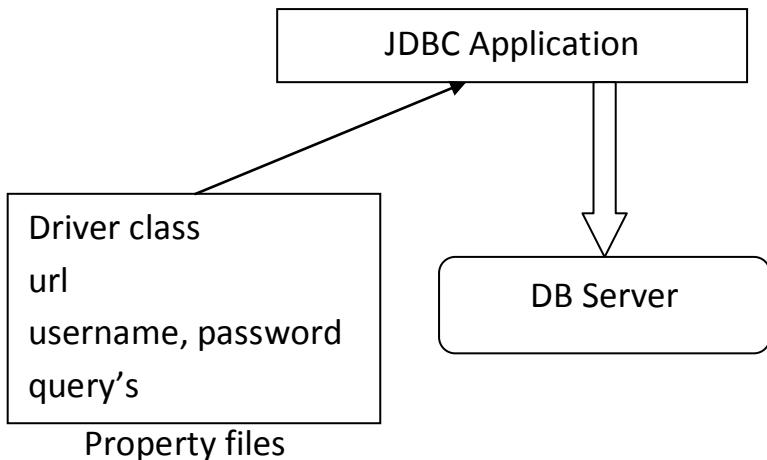
```
ResultSet rs = stmt.executeQuery("select * from product");
ArrayList list = new ArrayList();
While(rs.next()){
    Product p = new Product();
    p.setPid(rs.getInt(1));
    p.setPid(rs.getString(2));
    p.setPid(rs.getDouble(3));
    list.add(p);
}
System.out.println("After: " +list.size());
}
```

Hibernate guys has given the following three API's perform multiple row operations. They are:

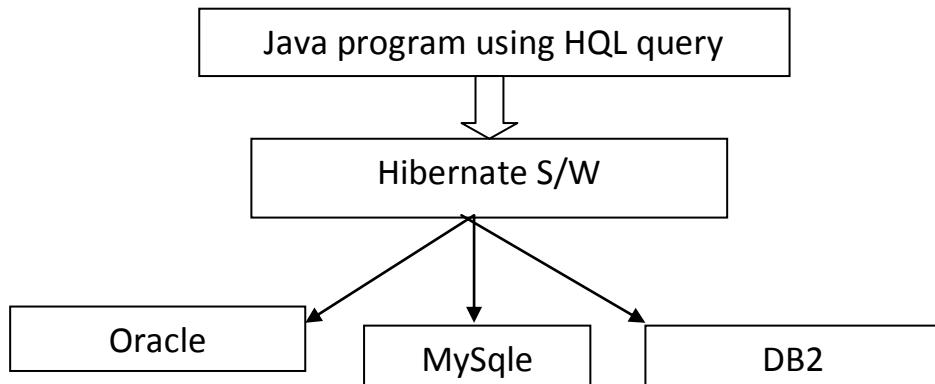
1. HQL API

2. Criteria API
3. Native sql API

When we develop jdbc application to communicate with any database server to remove the hard coding we use property files. As part of the property files we have provided driver class, url, username, password and queries. If we would like to communicate with other database servers we change the values from property files.



HQL queries as given as input to hibernate S/W it is the responsibility of hibernate to convert HQL query into corresponding SQL queries and send into data base server.



Hibernate guys has provided so many direct classes. As part of this dialect classes the code is provided to convert HQL query to corresponding sql queries.

The dialect classes are available in **org.hibernate.dialect** package. The following are some of the dialect classes. Oracle dialect, MySql dialect, SAP dialect etc.

The following are the features of sql.

1. HQL Queries fully object oriented.
2. HQL Queries supports inheritance and poly morphism.
3. HQL Queries are case sensitive.

If we want to write HQL queries instead of table names we have to use POJO calss names.

Instead of column names we have to use property names HQL queries derived from sql queries. The following is HQL queries derived from sql queries. The following is HQL and sql queries to retrieve all the records from emp table.

SQL> select * from emp;

Table name

HQL> from info.inetsolv.Employee

POJO class name

Develop a hibernate application which uses HQL queries to retrieve all the records from emp table.

***Procedure to use HQL in herberenate:**

1. Represent HQL query in the form of query object.
2. Send the query object to hibernate software by calling the list method.
3. Hibernate S/W returns an ArrayList object render the ArrayList object and display the output to client.
4. Query is an interface which is available as port of org.hibernate package. We can not create the object to query interface. We can create a reference variable and it holds implementation class object.

```
public class RetrieveData{  
    public static void main(String args[]){  
        // Standard code  
        Session hsession = sf.openSession();  
        Query query = hsession.createQuery(hql query);  
        List l = query.list();  
        ArrayList emplist = (ArrayList)l;  
        Iterator i = emplist.iterator();  
        while(i.hasNext()){  
            Employee e = (Employee).next();  
            System.out.println(e.getEmployeeNo());  
            System.out.println(e.getEmployeeName());  
            System.out.println(e.getEmployeeAddress());  
        }  
        hsession.close();  
    }  
}
```

When we call the list() method the following steps are carried out by list() method.

1. HQL query will be converted into corresponding SQL Query.
2. Hibernate S/W send SQL query to database server.
3. The database server executes select query and return the ResultSet object to hibernate S/W.
4. Hibernate S/W represents every record in the form of object and add it to array list object.
5. List() method converts ArrayList into supper class reference variable list and it returns it we have to get ArrayList object and display records to client.

When we trying to add any element to ArrayList object it will be converted into supper class object called as object. While retrieving data from ArrayList we need to typecast into appropriate object.

As part HQL queries we can add where conditions as well as order by clause, group by clause.

***using where clause as part of HQL:**

```
SQL> select * from emp where eno>2;  
HQL> from info.inetsolv.Employee where employee>2;
```

***using positional parameters in HQL:**

The following is an example of using positional parameters in HQL.

```
SQL> select * from product where price >? And price<?  
HQL> from product where price>? And price>?
```

Ex: String query = “from product where price>? And price<?”;

```
Query hqlquery = hsession.createQuery(query);  
hqlquery.SetDouble(0,2000);  
hqlquery.SetDouble(1,5000);  
ArrayList List = (ArrayList)hqlquery.list();
```

In hibernate the HQL query positional parameter index starts with 0.

If we do not supply the values to all positional parameters hibernate display an exception query exception.

We can use alias names as part of HQL query by using a keyword as.

Ex: String query = “from product as p where p.price>? and p.price<?”;

We can use order by as part of HQL queries.

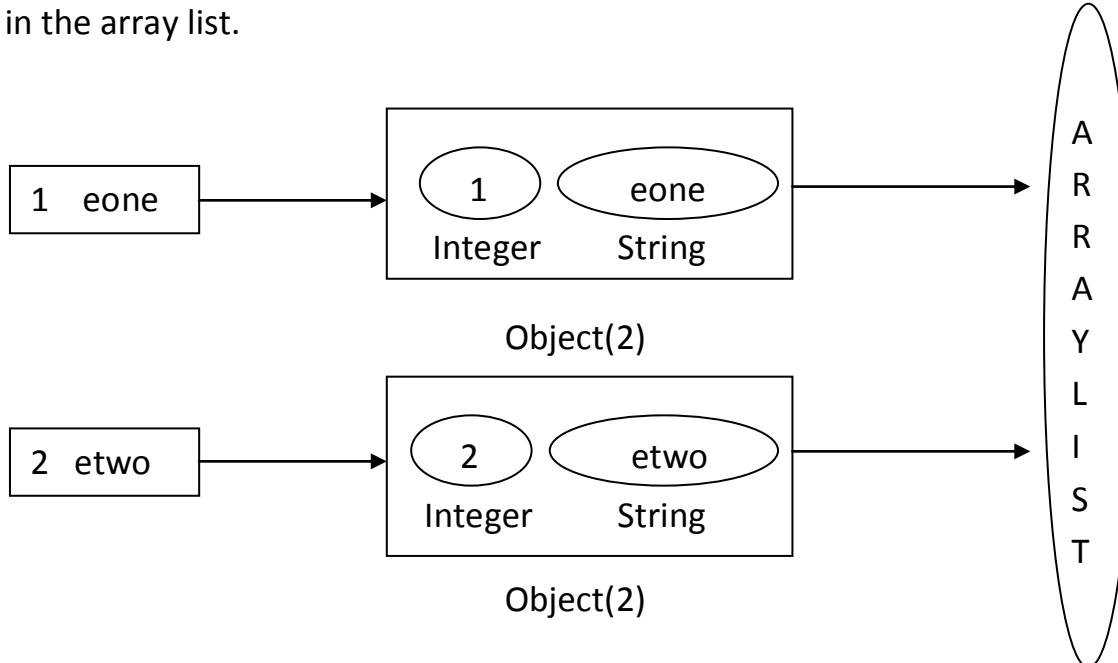
Ex: from product order by price desc

***Retrieving specific columns from employee table:**

The following example get employeeNo, employeeName columns only from the database server.

```
String query = "select employeeNO,employeeName from info.inetsolv.Employee";
Query hql query = hsession.CreateQuery(Query);
ArrayList List = (ArrayList)hqlQuery.List();
Iterator i = list.iterator();
While(i.hasNext()){
Object o[] = (object[]);
Next();
//code to display the data from arrays using index.
for(int j=0;j<0.lenth;j++){
System.out.println(o[j]);
}
}
```

The following diagram shows how hibernate list() method as converted the appropriate data in the array list.



Pro: Develop a hibernate application to retrieve employee name column only from emp table.

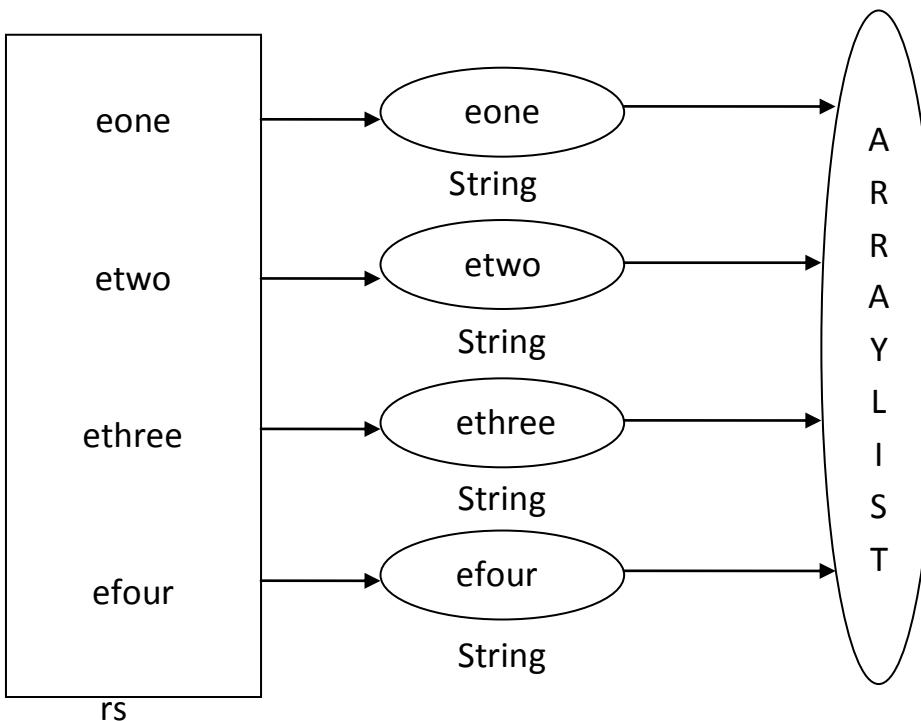
```
String q = "select employee name from employee";
Query que = hsession.CreateQuery(q);
ArrayList list = (ArrayList)Query.List();
Iterator i = list.iterator();
while(i.hasNext()){
Object o = i.next();
```

```

Integer ii = (Integer)o;
System.out.println(ii);}

```

When we have only one column in the select clause as part of the array list the data type will be added.



We can use aggregate functions in HQL queries. The following is an example using HQL to find number of records available in product table.

```

String q = "select count(*) from product as p";
Query que = hsession.CreateQuery(q);
ArrayList list = (ArrayList)que.List();
Iterator i = list.iterator();
while(i.hasNext()){
Object o = i.next();
Long l = (Long)o;
System.out.println(o);
}

```

We can join multiple tables and retrieve the records the following is an example of join tables.

```

Query HQLquery
ArrayList = (ArrayList)HQLquery.List();
Iterator := list.iterator();
while(i.hasNext()){

```

```

Object o[] = (object[])i.next();
for(int j=0;j<0.length;j++){
Employee e = (Employee) o[0];
Product p = (Product) o[1];
System.out.println(e.getEmployee NO() + "\t");
System.out.println(e.getEmployee Name() + "\t");
System.out.println(p.getpid() + "\t");
System.out.println(p.getName() + "\t");
}
}

```

By using HQL we can perform updating, deletion and insert records into a database server. To send the update query we use a method executed update().

```

Transaction tx = hsession.beginTransaction();
String query = "update info.inetsolv.Employee set employee Address!=
Where employee No = ?";
Query hqlQuery = hsession.createQuery(Query);
hqlQuery.setString(0,"xyz");
hqlQuery.setInteger(1,1);
int no = hqlQuery.executeUpdate();
System.out.println(no);
tx.commit();
hsession.close();
}
}

```

Note:

When we perform update, delete and insert operations. We must place the code in a transaction.

Hql will not support to insert record directly into table. if the data is already available in table we can copy from one table to another table.

```

Transaction tx = hsession.beginTransaction();
String Query = "insert into Employee(employeeNo,employeeName) select
pid, name from product";
Query hql Query = hsession.createQuery(Query);
int no = hql Query.executeUpdate();
System.out.println(no);

```

```
tx.commit();
hsession.close();
}
```

Named positional parameters:

When we write the hql Queries we can use positional parameters the problem with positional parameters is if somebody trying to read and understand it takes lot of time. Instead of positional parameters we use names. This will improve the readability of the queries.

Named positional parameters will not improve performance separately. This is because by default named positional parameters also uses prepare statements. The following is an example of named positional parameter.

Ex:

From product where price>:min value and price<:max value. The following is an example of using named positional parameters.

```
String Query = "from product where price>:minvalue and price<:maxvalue";
Query hql Query = hsession.CreateQuery(Query);
hql Query.setDouble("minvalue",2000);
hql Query.setDouble("maxvalue",6000);
ArrayList list = (ArrayList)hqlQuery.List();
```

In hibernate to retrieve the data from all the tables we can use hql query like from java.lang.object

Ex:

```
String Query = "from java.lang.object";
Query hql Query = hsession.CreateQuery(Query);
ArrayList list = (ArrayList)hqlQuery.List();
```

When the above list method is executed hibernate will send the queries to read the data from all the tables. The Array list object contains the POJO class object from table as well as emp tables.

Criteria API:

Criteria API is used to retrieve the records from the data base server the advantage of this API is we are going to represent the queries in the form of objects.

We can not use Criteria API to insert the record, update the record and delete the record.

The following is an example of using Criteria API to retrieve the records of product table.

Ex:

```
//standard hibernate code
Session hsession = sf.open Session();
Criteria c = hsession.Create Criteria(Product.class);
ArrayList list = (ArrayList)c.list();
Iterator i = list.iterator();
while(i.hasNext()){
    Product p = (Product);
    next();
    System.out.println(p.getId());
    System.out.println(p.getName());
    System.out.println(p.getPrice());
}
```

To work with Criteria API we must follow the following two steps:

1. Represent the Query in the form of Criteria object.
2. Send the Criteria to data base server by using a method list();

Hibernate guys has given predefined classes to add the restrictions. They are available as part of org.hibernate.criteria package some of the important classes in that are restrictions,order, property etc.

The following is code to add restrictions to criteria.

```
Criteria c = hsession.create Criteria(Product.class);
c.add(Restrictions.ge("price",2000d));
c.add(Restrictions.le("price",4000d));
```

The Restrictions class contains couple of static factory methods. The internal code of these methods add() the where conditions to the Query as a developer we are responsible to use these methods based on the requirements.

To sort the records based on Criteria API they have provided predefined classes like order. This contains the methods like asc, desc.

Ex:

```
Criteria c = hsession.Create Criteria(product.class);
c.addOrder(order.desc("price"));
```

To retrieve the specific columns from the data base table we use projections class.

Ex:

```
Criteria c = hsession.Create Criteria(product.class);
c.setProjection(Projections.property("name"));
```

When we run the above java program the Array list object contains the columns corresponding data type.

adding multiple projections:

```
Criteria c = hsession.Create Criteria(Product.class);
ProjectionList pl = projections.ProjectionList();
pl.add(Projections.property("name"));
pl.add(Projections.property("pid"));
c.setProjection(pl);
```

Note:

Criteria API is not best suitable for the real time projects and to develop complicated Queries for example using the functions like di code, nul1, nul2 etc.

Native SQL API:

The main advantage of Native SQL is we can write data base specific Queries as part of hibernate. By using Native SQL API we can perform the operations like insert Query, update, delete, retrieve and call the procedures and etc.

The following is an example of calling the procedure by using Native SQL.

Ex:

```
Transaction tx = hsession.beginTransaction();
SQLQuery Query = hsession.createSQLQuery("{call myproc}");
Query.executeUpdate();
tx.commit();
```

How do we use hibernate in web based applications:

How do we use hibernate in struts:

1. Create a web based application.

2. Copy all the hibernate related jar files into project lib folder.
3. Copy hibernate configuration file, POJO classes, hbm files and hibernate configuration files into classes folder.
4. Create hbm file and servlet to capture data and store data.

Generators:

In all the applications which we developed as of new end user is suppose to enter the primary key values.

It is not recommended to ask the end user to enter to enter the primary key values. It's always recommended to generate primary key values by the application. This resolves the problem of end user remembering the primary keys.

By using JDBC we have to write code to generate primary key values. If we use hibernate internally it contains a logic to generate the primary key values.

Approach1:

In this approach we find the maximum values from corresponding table and increment by one and use it as primary key value to find the next primary key value we can use the following query.

Ex:

```
Select max(eno) + 1 from emp;
```

This logic can be used in any data base server.

Approach2:

Oracle data base server supports a feature sequences to generate the primary key values to create the sequence we use the following query.

Ex:

```
create sequence empseq
min value 1
max value 99999
increment by 1
start with 1;
```

To see all the available sequences in the data base server. We use the following query.

```
SQL> select * from user_sequences;
```

Approach3:

In mysql there is a feature auto increment. If we use it mysql itself will increment the primary key values.

To use auto increment at the time of table is created we must specify auto incrementation.

Ex:

```
create table product(pid int(5) primary key auto_increment,  
name varchar(20),  
price decimal(10));
```

To insert the values we use the following query.

Ex: insert into product(name,price) values('abc',445);

auto increment will write in mysql and DB2. It does not work in oracle.

Hibernate has given set of predefined classes to deal with primary key values. These classes are available in a package. "org.hibernate.id"

The following are some of the classes of hibernate generators.

1. Assigned
2. Sequence generator
3. Increment generator
4. uuid generator
5. guid generator etc.

Hibernate guys has given an interface identifier generator this contain a method generator.

Identifier generator
Serializable generate(.....)

All the above generator classes must provide implementation to an interface identifier generator. As part of the generate method the logic is provided to generate the primary key value. This method returns primary key value in the form of serializable object.

As part of the hbm file there is a tag generator. This specify the generator which has to be used by hibernate.

EX:

```
<generator class = "org.hibernate.id .Sequence Generator"/>
```

Hibernate guys has provided short names for every generator. We can use the short names on be half of class names. The following are some of the short names. Assigned, sequence, increment, native, hilo, uuid etc.

The following is an example of using short names.

```
<generator class = "sequence"/>
```

Before we use a generator we have to check the following two steps.

1. Weather this generator can be used in the corresponding data base server or not.
2. We need to check weather the generator supports specific data type or not.

Assigned generator:

When we use assigned generator hibernate expect us to supply primary key value by default when we generator hbm and POJO classes generator is assigned with assigned value.

In the hbm file we can use generator with hibernate class name or short name.

```
<id name = "pid" type = "integer">
    <column name = "pid"/>
    <generator class = "org.hibernate.id.Assigned"/>
</id> // product.hbm.xml
```

OR

```
<id name = "pid"/>
    <column name = "pid"/>
    <generator class = "assigned"/>
</id>
```

In the hbm file if we does not supply the generator tag by default hibernate consider assigned generator.

If we supply a wrong class name or invalid generator class name hibernate throw an exception could not instantiate id generator.

Increment generator:

When we use this generator it will increment the primary key value by one based on existing primary key value. This algorithm internally uses the existing primary key value.

When we use this algorithm it will use the following query to find the maximum primary key value.

Ex: select max(pid) from product

The following is the configuration of increment generator.

```
<generator class = "org.hibernate.id.Increment Generator">
</generator>
<generator class = "Increment">
</generator>
```

Increment generator can generate the primary key values for short, long, integer data types only.

We can use increment generator in mysql data base server also.

Identify generator:

This generator can be used only in mysql data base server to use this generator compulsory the table must be auto increment. The following is the configuration of identify generator.

```
<generator class = "org.hibernate.id.identity generator"/>
```

OR

```
<generator class = "identity"/>
```

(sequencehilo):

Seq hilo: This generator uses both sequence as well as hilo value to generate a primary key value. This is same as hilo generator. this uses the sequence instead of table. the following is the configuration for sequence hilo generator.

```
<generator class = "seqhilo">
    <param name = "sequence">pidseq</param>
    <param name = "mar_lo">5</param>
</generator>
```

Uuid/guid: These generators generate the primary key value based on the IP address of the system and the start up time of JVM and convert it into 32 bit hexadecimal number and store in data base server.

To work with guid/uuid generators the primary key value data type must be var char with minimum size of 32 bit. The following is the configuration of generator.

```
<generator class = "uuid"/>
```

Instead of predefined generator we can use our own generator also to get the primary key values.

To develop our own generator class it must provide implementation to interface identifier generator. In this we need to provide the implementation to generator method.

Ex:

```

public class OurOwnGenerator implements Identifier Generator{
    public serializable generator(SessionImplementor session, object object){
        int eno = 0;
        ResultSet rs = session.getbatcher().PrepareBatchStatement("select
max(eno) from emp").executeQuery();
        if(rs.next()){
            eno = rs.getInt(1);
            eno = eno + 1;
        }
        return eno;
    }
}

```

To use this generator in hibernate we have to configure it in hbm file as show below.

Ex: <generator class = "info.inetsolv.ourown Generator"/>

When we use hibernate in form based applications we have provided the configure method and build session factory method as part of service method.

Single ton design pattern:

The main purpose of single ton design pattern is it make sure that any work execute only once or it make sure that the object is created to a class only once.

The following Cone class makes sure that it creates only one object for multiple people.

```

public class Cone{
    private static Cone c;
    Static{
        System.out.println("creating Cone object only once");
        c = new Cone();
    }
    public static Cone create Cone object(){
        return c;
    }
    Private Cone(){
        System.out.println("Cone object is created");
    }
}

```

To get the Cone class object we use a method created Cone object

```

public class MyApp(){
public static void main(String args[]){
Cone c1 = Cone.create Cone object();
Cone c2 = Cone.create Cone object();
System.out.println(c1);
System.out.println(c2);
}
}

```

The following is hibernate single ton design pattern. This class make sure that session factory is created only once.

```

public class Hibernate SessionFactorySingleTon{
private Static SessionFactory sf = null;
Static{
    Configuration cfg = new Configuration();
    cfg.configure();
sf = cfg.buildSessionFactory();
}
public Static Session Factory getSessionFactory(){
    return sg;
}
private HibernateSessionFactorySingleTon(){
}
}

```

As part of a servelet we get the session object directly from single ton design pattern.

Ex: public class StoreProductServlet extends HttpServlet{
 public void service(.){
 Session hsession = HibernateSession factory single ton . getSession();

 hsession.close();
 }
}

The following is an example hibernate temple design pattern. By using this any body can perform the some operation by using one line.

Ex:

```

public static void save(object o){
Session hsession = Hibernate session factory.getSession();
hsession.beginTransaction();
hsession.Save(o);
hsession.getTransaction().commit();
Hibernate session factory.close Session();
}
}

```

Develop a servlet to get all the records from product table and display to the client.

Named Queries:

Named Queries will improve the performance of java application.

When we develop a web based application to retrieve the records from product table and display to the client by using HQL Queries every time it try to convert HQL Query into corresponding SQL Query. We can improve the performance of by using named Queries.

Procedure to use Named Queries:

1. Configure Named Queries as part of hbm files.

Ex: <hibernate_mapping>

```

<class--->
-----
</class>
<query name = "gpd">from info.inetsolv.product</query>
</hibernate_mapping> //product.hbm.xml

```

2. From the java application to use the Named Queries use a method.getNamedQuery()

```

Query query = hsession.getNamedQuery("gpd");
ArrayList al = (ArrayList)query.list();
Iterator i = list.iterator();

```

When we are using Named Queries we can supply named positional parameters also.

Ex: <query name = "gpd">from info.inetsolv.product where name =: name</query>

Named native SQL we can configure native SQL also as part of hbm files. To configure native sql we use a tag <sql_query>

Ex:

```
<hibernate_mapping>
```

```

<class----- >
-----
</class>
<sql_query>name = "hsql"> select * from product </sql_query>
</hibernate_mapping>

```

To call named sql from java application we use a getNamed Query this returns query object. When this send to data base server we getArrayList object with objectArray as the values.

Ex:

```

Query query = hsession.getNamedQuery("hsql");
ArrayList al = (ArrayList)query.list();
Iterator i = al.iterator();
while(i.hasNext()){
    object o[] = (object[])i.next();
for(int j=0;j<=0.length;j++){
    System.out.println(o[j]);
}
}

```

Instead of object Array if we want to return a POJO class object we use return tag.in.hbm files as shown below.

```

<sql-query name = "hsql">
    <return class = "info.inetslov.product"/> select * from product </sql-query>

```

As an alternative to connection method hibernate guys has provided do work method.hibernate guys has given interface work. In this there is a method execute. We provide any JDBC code has part of execute method.

```

public class JDBCWork implements works{
    public void execute(connection con){
        CallableStatemet cstmt = con.PrepareCall("{call myproc}");
        cstmt.Execute();
    }
}

```

When we write the business logic in the procedure they will improve the performance. The disadvantage of this approach. If we change the data base server the application will not work as a java programs we recommend to write the business logic in java application.

Hibernate is not the best solution when we write the business logic in procedure. It's not recommended to use procedures in the projects which uses hibernate. We are seen three approaches of calling procedures from hibernate.

1. By getting the connection from session object.
2. By using native SQL application.
3. By using do work method.

When we use do work method hibernate internally uses callback mechanism to call execute method the advantage of this is we are not opening the connection we are not closing the connection. A procedure can write multiple records to achieve this we have to use cursors in procedure.

Cursors are a temporary memory area. Where the records will be stored the cursor object will be created by oracle. When ever select query is executed in oracle they are two types of cursors available. They are:

1. Implicit cursors
2. Explicit cursors

The following procedure returns a cursor/ResultSet object create or replace procedure MyProc(rs out sys_refcursor)

As

Begin

open rs for select * from product;

end myproc;

/

Following java application to call the procedure and display to the client.

```
public class GetRecords{  
    public static void main(String args[]){  
        //standard jdbc code  
        CallableStatement cstmt = con.PrepareCall("{call myproc(?)}");  
        cstmt.Execute();  
        ResultSet rs = (ResultSet) cstmt.getObject(1);  
        //code to display the records  
    }  
}
```

Hibernate guys are giving a way to call procedure which takes a parameter refcursor. Configure procedure in hbm file as shown below.

```

<hibernate_mapping>
<class ----- >
-----
</class>
<sql_query name = "cp" callable = "true">
<return class = "info.inetsolv.product"></return>
{call myproc(?)}
</sql_query>
</hibernate_mapping>

```

To call the procedure from java we use named queries as shown below.

```

Query query = hession.getNamedQuery("cp");
ArrayList al = (ArrayList)query.list();
Iterator i = al.iterator();
while(i.hasNext()){
    product p = (product)i.next();
}
}

```

We have to follow limitations in hibernate to call the procedure.

1. The first parameter procedure must be out parameter and the data type must be ref cursor.
2. When we are configuring the procedure in we must use on attribute callable st.

When we specify callable true hibernate register. The first parameter as out parameter we must use native sql to call the procedure from hibernate application.

If you want to search the records based on given product id you have to supply a product id as parameter to procedure. We can call that the configure in hbm file as shown below.

We can develop a hibernate software to communicate with multiple data base servers from a single project. Procedure to use hibernate with multiple data base servers for the same project. Create multiple hibernate configuration files.

Ex:

```

hibernate.cfg.xml
hibernate.mysql.cfg.xml

```

To use multiple data base servers in the project we use multiple configuration files and multiple session factory objects. In the projects we do this work only once by using a single ton design pattern only.

```
public class HibernateSessionFactory{  
    private static SessionFactory osf = null;  
    private static SessionFactory msf = null;  
    static{  
        Configuration ocfg = new Configuration();  
        ocfg.Configure();  
        Configuration mcfg = new Configuration();  
        mcfg.Configure("hibernate.mysql.cfg.xml");  
        ocfg.buildSessionFactory();  
        mcfg.buildSessionFactory();  
    }  
    public static SessionFactory get mysqlsessionFactory{  
        return ocfg; }  
    public static SessionFactory get mysqlsessionFactory{  
        return mcfg; }  
    private HibernateSessionFactory(){}  
}
```

In the java application we get the required Session Factory object and perform the operations.

Ex: Sun Micro System has released JPA API to interact with data base servers they are released this API as part of JEE standard API.

The following is the classes and interfaces as part of JPA.

<u>Classes</u>	<u>Interfaces</u>
Persistence	

Hibernate guys provided the implementation to JPA API.

Create a java project and add JPA capabilities. There are so many people who are providing the implementation same of them or top link, hibernate and open JPA and etc. we can use any jar files to develop JPA application. When we add JPA capabilities it has created an xml file whose name is persistence.xml in this we have specified persistence unit name.

Generate an Entity Beans (POJO classes). Develop a java application to store the data into employee table.

```
public class StoreEmp{
```

```

public static void main(String args[]){
EntityManagerFactory emf = Persistence.CreateEntityManagerFactory("JPAORACLEP");
EntityManager em = emf.CreateEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
Emp e = new Emp();
e.setEno(1);
e.setEno(2);
e.setEno(3);

em.Persist(e);
tx.commit();
em.close();
}
}

```

The top link provides the implementation of JPA API to work with top link we are to download the top link jar files and set the class path. When we are developing a project at the design phase we try to develop two types of design.

1. Data base design
2. Java design

As part of data base design we will decide the names of the tables and columns of the tables and etc. At the time of developing the project developer will not create any tables. In most of the projects we have to capture multiple address to an employee.

Eno
 Name

Address:

Street	city	state
Am pet	HYD	AP
SR Nagar	HYD	AP

ADD another address

To implement above requirement we got the following design.

Design 1:

Eno	Name	Street	State	City
1	Naidu	Am pet	AP	HYD

As part of design we are adding two tables. Emp table and address table in the Emp table Eno is a primary key. In the address table Eno, Address no are forming the primary key.

Design 2:

Eno(PK)	Name	salary
1	Naidu	4000

Eno	Addno	Street	State	City
1	1	Am pet	AP	HYD

In the design2 the address table contain Eno, addressno as primary keys. If more than one column in valued in forming a primary key we call it as composite primary key. Hibernate will not give the best performance if the table contains composite primary key. Hibernate give the best performance if it's having only one primary key.

Design 3: In this design we have two tables. They are emp and address.

In the address table we will try to have a new column which acts as primary key.

Eno(PK)	Name	salary
1	Naidu	100000
2	Malli	200000

Aid	Eno(PK)	addno	Street	State	City
1	1	1	APet	HYD	AP
2	2	1	APet	HYD	AP
3	2	2	APet	HYD	AP

Generally by using hibernate we should be able to create the tables. but according to the project design first we are creating tables from the tables we are creating POJO

classes. Creating the POJO classes from the table is called as hibernate reverse engineering.

There are some situations where we need to develop hibernate applications based on legacy data base servers.

****What are legacy data base servers?**

The data base server which uses composite primary key's is called as legacy data base servers. Procedure to develop hibernate applications based on legacy data base. Create the following the two tables.

Table 1: create table emp(eno number(5) primary key,

Name varchar2(20), salary number(10,2);

Table 2: create table address(eno number(5), addrno number(5),

State varchar2(20), primary key(eno, addrno));

When we generate hbm files and POJO classes we have observed that 3 POJO classes and two hbm files the POJO classes are:

Emp.java

AddressId.java

Address.java

We have not observed any change in emp POJO class. This is because this table is having only one primary key. We have observed that address table is created. Two POJO classes:

1. Composite primary key class(AddressId)
2. Regular POJO classes

The composite primary key class contain two properties this is because two columns involving in forming composite primary key.

```
public class AddressId{
```

```
    Integer Eno;
```

```
    Integer addrno;  
  
    //setters and getters  
  
}
```

The following is address.java which uses composite primary key class as a property.

```
public class Address{  
  
    AddressId id;  
  
    String street;  
  
    String city;  
  
    String state;  
  
    //provide setters and getters }
```

We are generated two hbm files. They are:

1. Emp.hbm.xml
2. Address.hbm.xml

We have not observed any changes in Emp.hbm.xml files. We have observed lot of changes in address.hbm.xml file with respect to composite primary key.

```
<hibernate_mapping>  
  
<class name = “info.inetsolv.Address” table = “Address”>  
  
<composite-id name = “id” class = “info.inetsolv.AddressId”>  
  
<key-property name “Eno”>  
  
<column name = “Eno”/>  
  
</key-property>  
  
<key-property name = “addno”>  
  
<column name = “Addno”/>  
  
</key-property></composite-id>
```

```
<property name = "street"/>  
<property name = "city"/><property name = "state"/>  
</class>  
  
<hibernate_mapping>
```

The following java application to store the data into data base server creating the emp obj.

1. AddressId obj
2. Address obj
3. Call save method for emp.address

To establish the relationship between tables we do it based on the data available in the tables. They are:

1. One to many
2. Many to one
3. One to one
4. Many to many

To establish the relationship between tables we do it based on the data available in the tables.

One to many relationship:

To identify the relationship between the two tables we use the data based on the data available in the tables develop a hibernate application which uses one-to-many relationship b/w emp and address table. The following of the two tables are created by emp and address.

```
create table emp(Eno number(5), Name varchar2(20), salary number(10,2));
```

```
create table emp add primary key(Eno);
```

```
create table address(Aid number(5), Eno number(5), Addno number(5), city  
varchar2(10));
```

```
alter table address add primary key(Aid);
```

```
create table address(Aid number(5) primary key, Eno number(5) references emp(Eno)
```

when we generate hbm files and POJO classes we got two hbm files and POJO classes. The following is the POJO class from emp table.

```
public class emp{  
    Integer Emp;  
    String name;  
    Double salary;  
  
    Set address = new HashSet();  
  
    //provide setting and getters  
}
```

```
alter table address add  
foreign key(Eno) references  
emp(Eno);
```

The following is the configuration of the Emp.hbm.xml represents the many relationship.

```
<hibernate_mapping>  
  
<class name = "info.inetsolv.Emp">  
  
    <id name = "Eno"/>  
  
    <property name = "name"/>  
  
    <property name = "salary"/>  
  
    <set name = "address" inverse = "true">  
  
        <key>  
            <column name = "ENO"/>  
        </key>  
  
        <one-to-many class = "info.inetsolv.Address"/>  
    </set>  
  
</class>  
  
<hibernate_mapping>
```

The following is the POJO class for Address table.

```
public class Address{  
    Integer aid;  
    Emp emp;  
    Integer addno;  
    String street;  
    String city;  
    String state;  
    // provide setters and getters  
}
```

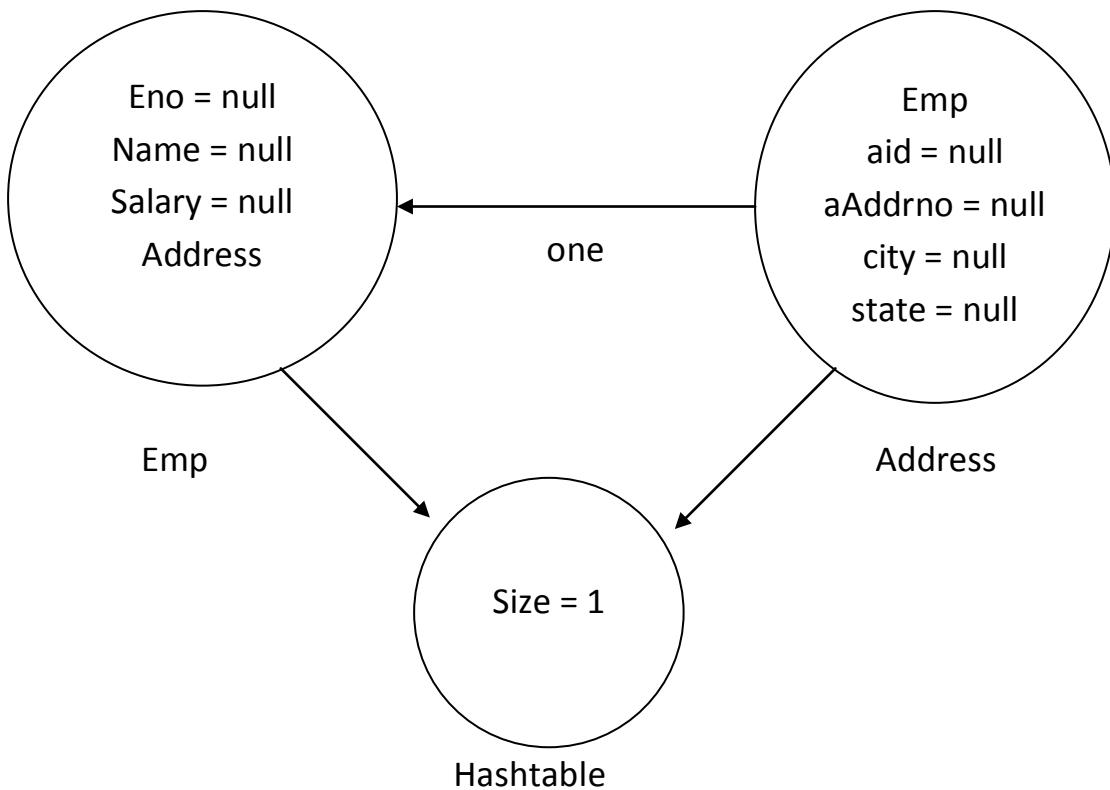
The following hbm file for address table.

```
<hibernate_mapping>  
<class name = "info.inetsolv.Address">  
    <id name = "aid"/>  
    <many-to-one name = "EMP" class = "info.inetsolv.Emp">  
        <column name = "ENO"/>  
    </many-to-one>  
    <property name = "addrno"/>  
    <property name = "street"/>  
    <property name = "city"/>  
    <property name = "state"/>  
</class>  
</hibernate_mapping>
```

Generally IDE recognize the relationship between tables and generate hbm files and POJO classes. Hibernate understand the relationship between the tables based on the tags which are available in hbm file. To represent many relationship in hibernate we use collection objects. The following are collection objects of hibernate.

List Set Map Bag

These collection objects represents many relationships we use hibernate from the java side we have to create the objects store the data into objects and establish circular relationship.



The following java application to store the data into emp as well as address table.

```
public class store{  
public static void main(String str[]){  
Session h = HibernateSessionFactory.getSession();  
Transaction tx = h.beginTransaction();  
Emp e = new Emp();  
e.SetEno(1);  
eSetName("Naidu");
```

```
e.SetSalary(10,000);
Address a = new Address();
a.SetAid(1);
a.SetEmp(e);
a.SetAddno(1);
a.SetCity("HYD");
a.SetState("AP");
Set s = e.getAddress();
s.add(a);
h.Save(e);
h.getTransaction.Commit();
HibernateSessionFactory.CloseSession();
```

When we call save(e) hibernate can find the relationship between the tables and store the data into dependent on tables to do this we must specify two attributes inverse and cas code.

```
<Set ----- inverse = "true" cascade = "all"/>
```

Develop a java application to retrieve the data from emp table who emp no is 2 (we would like to retrieve the corresponding address details). Hibernate checks two strategies to retrieve the record from data base server.

1. Lazy loading -----> Aggrisive (eager) loading.

****What is lazy loading?**

When ever we perform can operation on one table and table has relationship with other tables and then we called a load method it will retrieve the data from that table only.

Hibernate retrieve the records from child table when even any operation is carried out of any other table. By default hibernate uses lazy loading(true).

****What is Aggressive loading?**

When ever we call a load method on a parent table it will try to retrieve record form child table also even though user doesn't access any properties. How to make hibernate as aggressive we have to configure lazy = "false" in hbm file of parent table.

```
<set ----- inverse = "true" lazy = "false">
</set>
```

The following is the example of retrieve the data from data base server by using the load() method which is loading the relationship.

```
public class Retrieve{  
    public static void main(String args[]){  
        Session s = HibernateSessionFactory.getSession();  
        Emp e = new Emp();  
        e.getEno();  
        e.getName();  
        e.getSalary();  
        Set address = e.getAddress();  
        System.out.println(address.Size());  
        Iterator i = addresses.iterator();  
        while(i.hasNext());  
        Address a = (Address)i.next();  
        System.out.println(a.getAid());  
        System.out.println(a.getAddno());  
        System.out.println(a.getCity());  
    }  
    HibernateSessionFactory.getSession();  
}
```

As part of POJO classes and hbm file instead of set we can use only hibernate type List, map, set, bag.

Ex: We are using the list in the emp table is shown below.

```
public class Emp{  
    Integer eno;  
    String name;  
    Double salary;  
    List address = new ArrayList();  
}
```

The following configuration of Emp.hbm.xml

```
<hibernat_mapping>  
<class name = “info.inetsolv.emp”>  
<id name = “Eno”/>  
<property name = “name”/>  
<property name = “salary”/>  
<list name = “address” inverse = “+ve” cascade = “all”>  
<key column = “Eno”/>
```

```

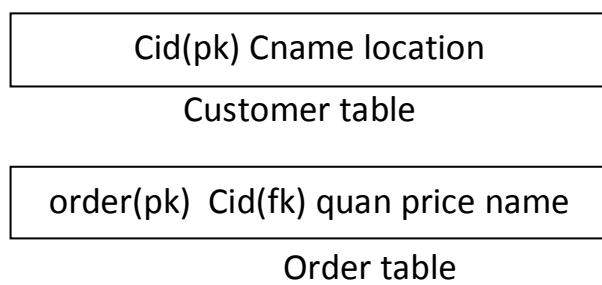
<index>
</index>
<one-to-many class = "info.inetsolv.Address"/>
</list>
</class>
</hibernat_mapping>

```

By default hibernate uses inverse = “false” if inverse = “false” hibernate will not check the bi-directional relationship between tables. In this scenario if we try to send multiple insert queries and update query. In case of inverse = “true” he check the bi-directional relationship and store the data by using insert queries.

Cascade attribute takes multiple values. They are none, all, save-update, delete.

1. Cascade specifies when ever we perform any operation on parent table it perform the operation on child table also. By default in hibernate cascade attribute value is name.
2. When we specify cascade = none when ever we carry out any operation on parent table it does not carry out any operation on child table.
3. When we specify cascade = “save-update” when ever we perform save operation on parent table it will perform save operation on the child table also.
4. Cascade delete when we specify cascade delete when ever we delete the record from parent table we delete the record from child table also. Instead of specifying every operation we specify cascade = “all”. We would like to maintain the customer and order details.



many-to-one: We would like to maintain the tables for multiple employees we would like to have departments and there are departments with out employees to achieve we use many-to-one relationship.

Eno(pk) name Did(fk)

Emp

Did(pk)	name
---------	------

Dept

one-to-one relationship:

We would like to maintain one-to-one relationship between team and captain tables. The following are two takes which we are using to achieve this.

Tid(pk)	name	location
1	dc	'hyd'

Team

Tid(pk)	(pk)	Cname
1		Naidu

Captain

Note: When we create the above two tables and establish the relationships and generate hbm files and POJO classes IDE as un-necessarily created one-to-many relationship this is because of adding a foreign key relationship between captain and team table.

1. Create two tables team and captain with only primary key do not add the foreign in the beginning this is because if we had a foreign key IDE will generate one-to-many relationship.
2. Generate hbm files and POJO classes.
3. We have to add one-to-one relationship between team and captain by using properties

<hibernate-mapping>

 <class . . . >

 <one-to-one name = "captain" class = "info.inetsolv.captain" cascade = "all">

 </one-to-one>

```

</class>
</hibernate-mapping> //team.hbm.xml

```

Similar to team POJO class and hbm file, develop captain java and captain.hbm.xml

*Develop the hibernate application to store data in to data base server.

```

public class StoreTeamDetails{
public static void main(String[] args){
Session hsession = HibernateSessionFactory.getSession();
Transaction tx = hsession.beginTransaction();
Team t = new Team();
t.setTid(tid);
t.setName(teamName);
t.setLocation(location);
Captain c = new Captain();
c.setTid(tid);
c.setName(captain);
c.setTeam(t);
t.setCaption(c);
hsession.save(c);
tx.commit();
HibernateSessionFactory.closeSession();
}
}

```

many-to-many:

We would like to implement many-to-many relationship b/w student and course tables.

Sid(pk)	Sname
---------	-------

//student

Cid(pk)	Cname
---------	-------

//course

Sid(pk)	Cid(pk)
---------	---------

//student-course

To specify the many-to-many relationship b/w the tables we have to provide the following configuration in hbm files.

```

<set name = "StudentCourses" table = "Student_Course" CasCode = "all" inverse =
"true">
    <key column = "Sid"/>
    <many-to-many column = "cid" class = "info.inetsolv.course"/>

```

```

</set>      //student.hbm.xml
***The following is the java code to store data in the data base server.
public class StoreData{
public static void main(String args[]){
Session hession = HibernateSessionFactory.getSession();
Transaction tx = hession.beginTransaction();
Course c = new Course();
c.setCid(1);
c.setCname("java");
Student s1 = new Student();
s1.setSid(1);
s1.setSname("raju");
Student s2 = new Student();
s2.setSid(2);
s2.setSname("naveen");
setStudents = c.getStudentCourse();
Students.add(s1);
Students.add(s2);
c.setStudentCourses(students);
hession.save(c);
tx.commit();
HibenateSessionFactory.closeSession();
}
}

```

***** Spring *****

Framework is a piece of software. This resolves the commonly and repeatedly occurred problems in multiple projects.

Spring framework is **an open source** Java platform and it was initially **written by Rod Johnson** and was first released under the Apache 2.0 license in June 2003.

Spring is more flexible framework when we compared with struts. By using Struts 1.3 and 2.3 we can develop only web based applications. But by using spring we can develop web apps as well as Standard alone applications also.

We can use any of the following frameworks to develop the web based applications.

The y are :

- 1. Struts 2.Spring 3.Jsf 4.Web work 5.OpenSymphony

Spring framework is a piece of software which **contains the solutions** to commonly repeatedly occurred problems across multiple projects.

The following are the advantages of Frameworks like Struts and spring:-.

- Frameworks resolve the problems of identifying the architecture. Every framework like Struts and Spring is delivered with MVC2 architecture.
- If we use Servlets and Jsp's we have to develop our own Controllers. If we use frameworks like Struts and **Spring internally they came with Controllers**

Eg: **ActionServlet** in Struts

DispatcherServlet in Spring

- When we use any framework we no need to **use RequestDispatcher code** as well as we no need to hard code the resource path names. By using these frameworks we can configure them in the configuration files.
- If we use JDBC, Servlets, Jsp's to develop the form based applications we have to write huge amount of code to take care of **Server side validations** and displaying errors in the same form.
- By using JDBC, Servlets, Jsp's we have to provide huge amount of code to develop the **I18n applications**. (The programs which displays the output based on client regional Languages)
- The frameworks like Struts and spring delivered with set of **predefined tag libraries**. If we use Servlets and Jsp's we have to develop our own tag library which is difficult.
- When We use the frameworks like Struts and spring we can use **pdf/velocity/jsf as view components**.

Most of the experienced guys are develops the new frameworks.

Every Company uses their own frameworks to develop the projects .all these frameworks internally uses the other frameworks.

Benefits of Using Spring Framework:

Following is the list of few of the great benefits of using Spring Framework:

- Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.
- Spring is organized in a **modular fashion**. Even though the number of packages and classes are substantial, you have to worry only about ones you need and ignore the rest.
- Spring does not reinvent the wheel instead, it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, other view technologies.
- Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using JavaBean-style POJOs, it becomes easier to use dependency injection for injecting test data.

- Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over engineered or less popular web frameworks.
- Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.
- Lightweight **IoC containers** tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.
- Spring provides a **consistent transaction management** interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

The main feature of spring is **IOC**.

What is IOC?

IOC stands for Inversion of Control. A person 'x' acts as supposed to do a work instead of that person perform that tasks some other person 'y' completed the task and the result is enjoyed by 'X' person. We will call this as Inversion of Control.

Spring supports **AOP**. AOP stands for Aspect Oriented Programming.

The advantages of Aop are all the common code is place in aspect before the business logic method is executed spring call the Aspect. The aspect internally called the business logic method.

As per spring they have provided the transaction Manager aspect. He will take care about the transaction managing in Spring.

Spring framework follows mVc2 architecture and it is an open source.

Spring frame work is divided into couple of modules. The advantage of this approach is we can include only required modules in our project.

The module is a collection classes and interfaces and enumerations.

Spring framework is divided into six modules :

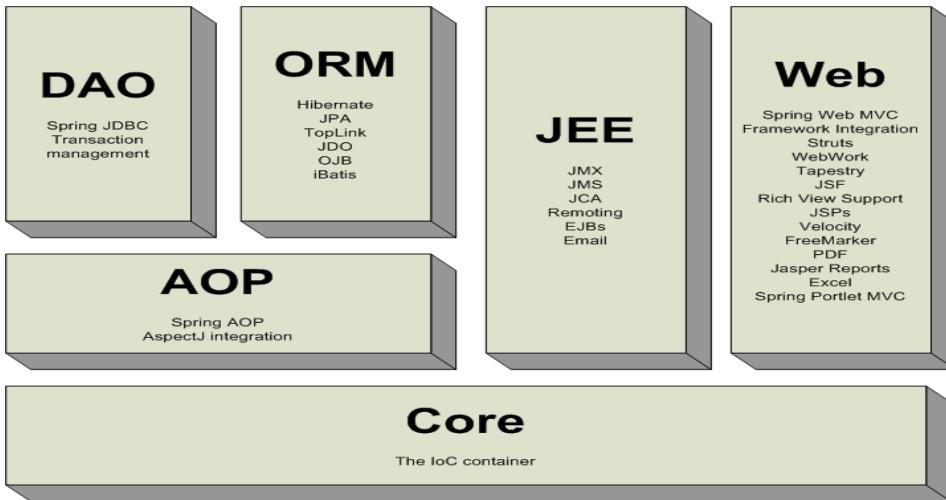
They are:

1.Core 2.DAO 3.ORM 4.AOP 5.JEE 6.WEB

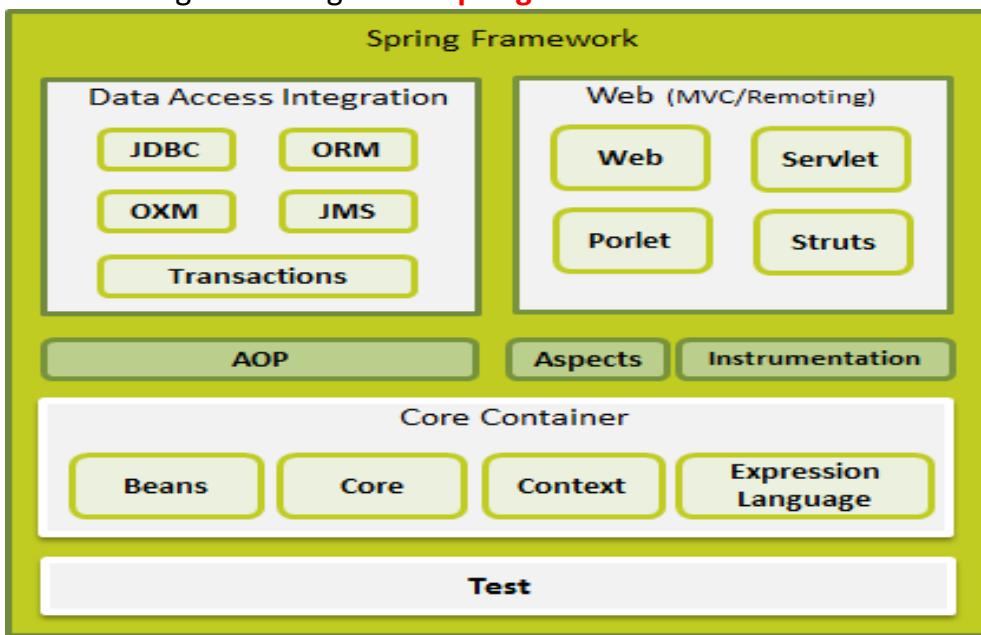
Sometime in the project we will use only core ,orm, aop only..

The following is the architecture of spring 2.0

spring 2.0 modules



The following is the diagram of **spring 3.0 modules**:



Core module:

As part of core module we have IOC.

The Core Container consists of the Core, Beans, Context, and Expression Language modules whose detail is as follows:

- The **Core** module provides the fundamental parts of the framework, including the IoC and Dependency Injection features.
- The **Bean** module provides BeanFactory which is a sophisticated implementation of the factory pattern.
- The **Context** module builds on the solid base provided by the Core and Beans modules and it is a medium to access any objects defined and configured. The ApplicationContext interface is the focal point of the Context module.
- The **Expression Language** module provides a powerful expression language for querying and manipulating an object graph at runtime.

Data Access/Integration:

The Data Access/Integration layer consists of the JDBC, ORM, OXM, JMS and Transaction modules whose detail is as follows:

- **The DAO module:** When we use DAO module we can reduce a lot of code in project. DAO module resolves the all **problems of database specific errors**.
- The **JDBC** module provides a JDBC-abstraction layer that removes the need to do tedious JDBC related coding.
- The **ORM** module provides integration layers for popular object-relational mapping APIs, including JPA, JDO, Hibernate, and iBatis.
- The advantage of spring orm is if we knowledge in one orm we can work any other orm module easily
- The **OMX** module provides an abstraction layer that supports Object/XML mapping implementations for JAXB, Castor, XMLBeans, JiBX and XStream.
- The Java Messaging Service **JMS** module contains features for producing and consuming messages.
- The **Transaction** module supports programmatic and declarative transaction management for classes that implement special interfaces and for all your POJOs.

Web:

The Web layer consists of the **Web**, **Web-Servlet**, **Web-Struts**, and **Web-Portlet** modules whose detail is as follows:

- The **Web** module provides basic web-oriented integration features such as multipart file-upload functionality and the initialization of the IoC container using servlet listeners and a web-oriented application context.
- The **Web-Servlet** module contains Spring's model-view-controller (MVC) implementation for web applications.
- The **Web-Struts** module contains the support classes for integrating a classic Struts web tier within a Spring application.
- The **Web-Portlet** module provides the MVC implementation to be used in a portlet environment and mirrors the functionality of Web-Servlet module.

Miscellaneous:

There are few other important modules like AOP, Aspects, Instrumentation, Web and Test modules whose detail is as follows:

AOP:

- AOP stands for Aspect oriented program. this resolves the problem of Oops.(inheritance between the multiple classes)
- The open source group people come with Programming model called AOP. The software AspectJ followed the Aop and developed the Aop module. Spring guys has developed some classes which follows the aop module.

- The **AOP** module provides aspect-oriented programming implementation allowing you to define method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
- The **Aspects** module provides integration with AspectJ which is again a powerful and mature aspect oriented programming (AOP) framework.
- The **Instrumentation** module provides class instrumentation support and class loader implementations to be used in certain application servers.
- The **Test** module supports the testing of Spring components with JUnit or TestNG frameworks.

JEE module in spring2.0:

JEE stands for java environment enterprise.

As part of jee module spring guys has supported **EJB and MailAPI**.

DependencyLookup:

If the resource of the application searches and gathers its dependent values from other resources of the application is called as dependency lookup. In dependency lookup resources pulls the values from other resources.

If values are assigned to variables only after calling the method explicitly is called as dependency lookup. The way the client app gathers **DataSource** object reference from registry software is called dependency lookup.

Dependency injection:

If underlying container or framework or server or runtime environment is dynamically assigning dependent values to resources of the application then its called **Dependency injection or IOC**.

In **Dependency injection** underlying container/framework dynamically pushes the values to resources..

Eg: The way ActionServlet writes the form data to FormBean class object comes under Dependency injection.

Spring supports three types of **Dependency injections**:

1. Setter Injection (By using setXxx(xxx) methods)
2. Constructor injection (by using constructors)
3. Interface Injection (by implementing special interfaces)

Any predefined or userdefined java class can be called as Spring bean.

The java class that contains only Persistence logic and separate that logic from other logics of application is called as **DAO**.

Spring supports POJO class . POJO class is nothing but a class which is not extending or any other class or interface.

Procedure to set up the first Spring application:

Steps:

1. Create a work space for spring application .

2. Create the java project .
3. Add the spring libraries to project by selecting add spring capabilities in MyEclipse. And add user library in the eclipse ide following way:
**Project-->properties----->JavaBuildpath----->Add library----->
UserLibrary----->new ---> give name for lib (spring)----->Add jars-->
Then add all spring jar files.**

For developing Spring app u have to create the following classes:

1. Spring bean class
2. Spring bean configuration class
3. Client application to get the bean class object

Developing Spring bean class:

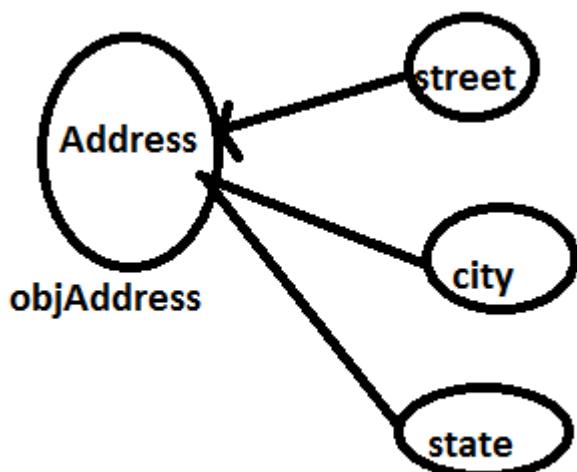
We have created the Address class with three properties

```
Public class Address {
    String street;
    String city;
    String state;
    //setter and getter for the above class like follow
    Public void setStreet(String street){
        this.street = street;
    }
    public String getStreet(){
        return street;
    }
}
```

we have created Address class object as follow as

Address objAddress = new Address();

we called a method **objAddress.setStreet("srnagar")** when this setter is called we have created a String object and given supplied input to that method. And it supply the value for instance variable.



```
objAddress.setStreet("srnagar");
objAddress.setStreet("hyd");
objAddress.setStreet("ap");
```

From above diagram we have observed that objAddress is dependent on String objects. The meaning this is Address object is using the String object.

when the setStreet() is executed it stabilizes the dependency between String and Address object. As the Setter() method is stabilizes the dependency between two objects we are calling this as **Setter injection**.

Instead of creating an object and stabilize the dependency, Spring container will take care of creating object and stabilize the dependency or injecting the dependencies We will call this as Inversion of Control or dependency injection.

According to Spring documentation Ioc is called as Dependency injection.

In Spring we will develop the pojo or spring bean class . If spring Container want to take care of Ioc then we have to configure that in Spring configuration file.(ApplicationContext.xml)

The following is example of spring bean configuration file:

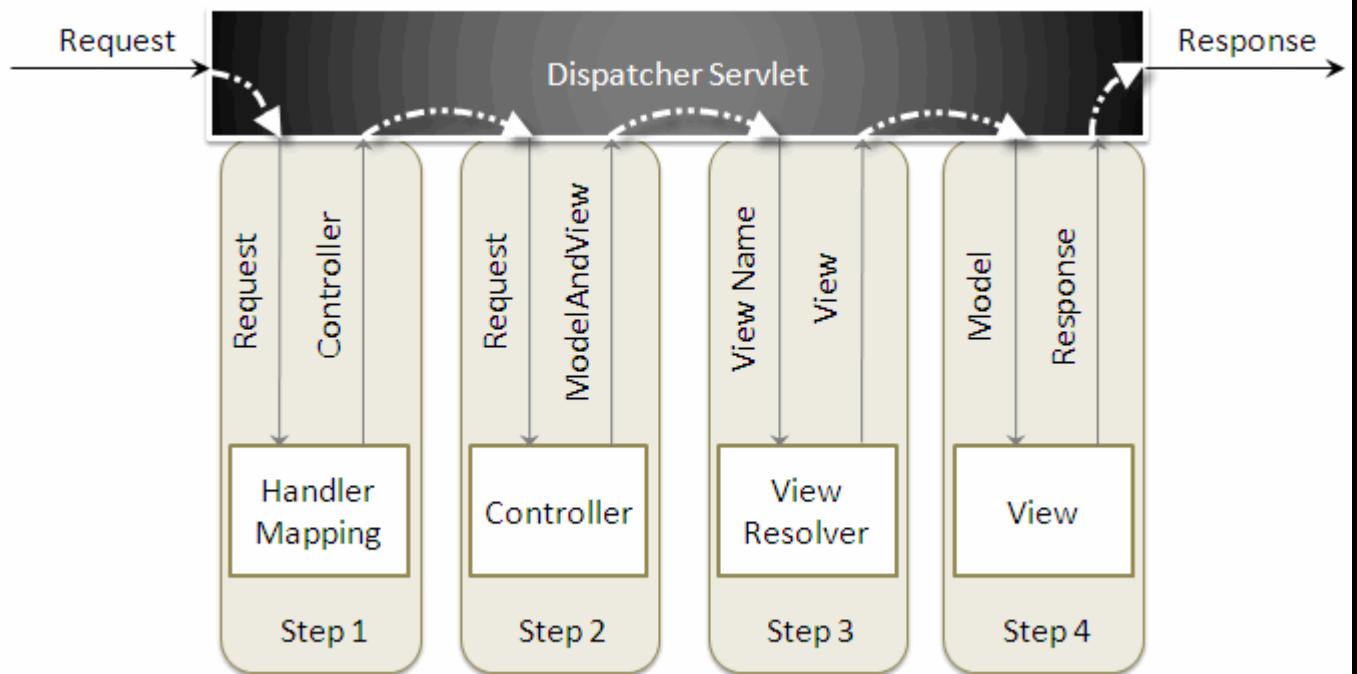
```
<beans>
    <bean id="" class="" lazy-init="">
        <property></property>
    </bean>
<beans>
```

Procedure to configure spring bean in Spring configuration file in MyEclipse:

Spring MVC Framework

Spring MVC helps in building flexible and loosely coupled web applications. The Model-view-controller design pattern helps in **separating the business logic**, presentation logic and navigation logic. Models are responsible for encapsulating the application data. The Views render response to the user with the help of the model object . Controllers are responsible for receiving the request from the user and calling the back-end services.

The figure below shows the flow of request in the Spring MVC Framework.

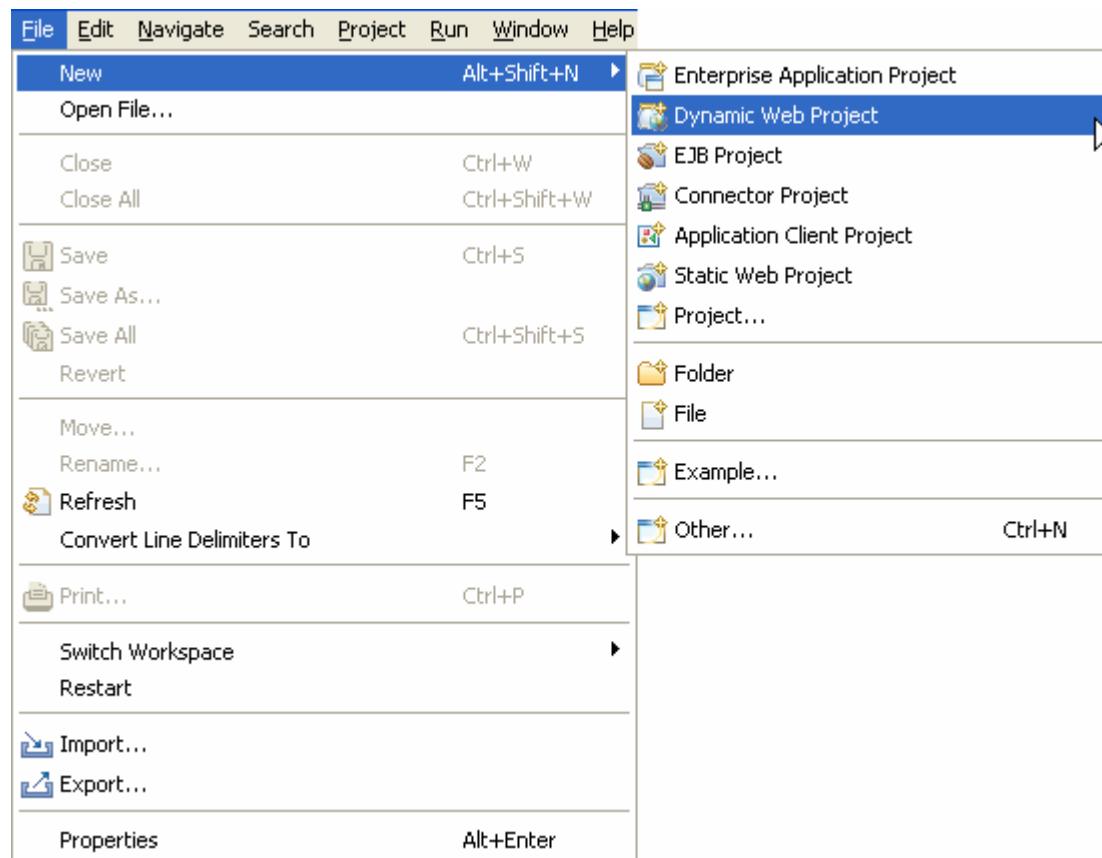


When a request is sent to the Spring MVC Framework the following sequence of events happen.

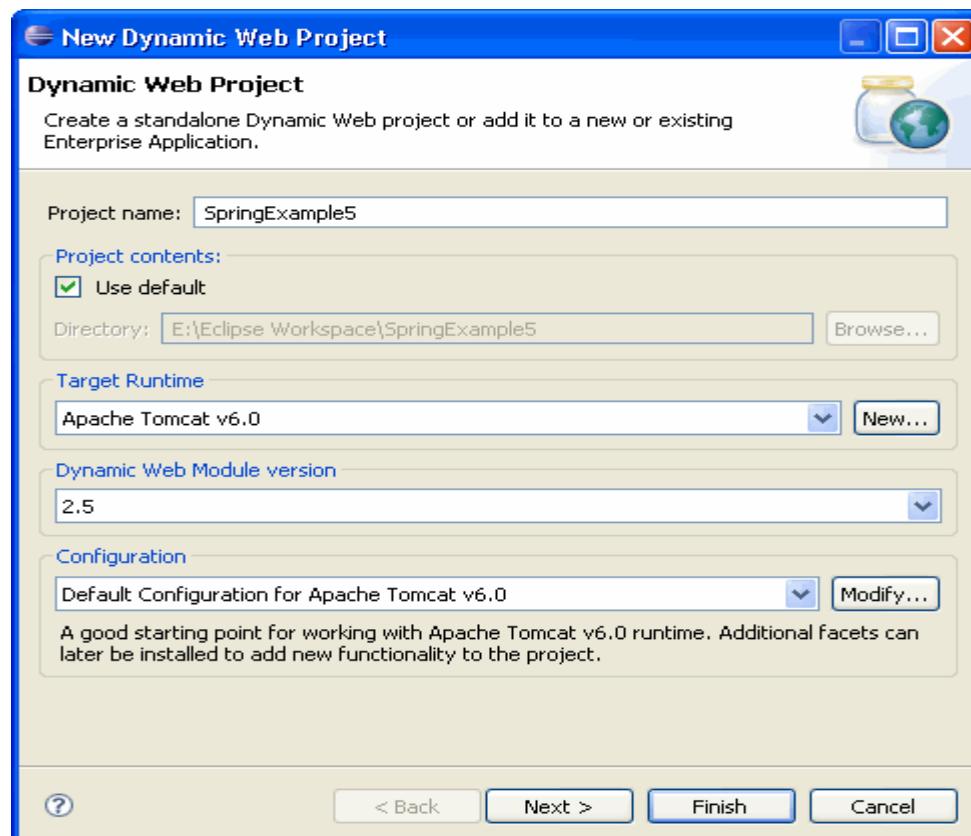
- The *DispatcherServlet* first receives the request.
- The *DispatcherServlet* consults the *HandlerMapping* and invokes the *Controller* associated with the request.
- The *Controller* process the request by calling the appropriate service methods and returns a *ModeAndView* object to the *DispatcherServlet*. The *ModeAndView* object contains the model data and the view name.
- The *DispatcherServlet* sends the view name to a *ViewResolver* to find the actual *View* to invoke.
- Now the *DispatcherServlet* will pass the model object to the *View* to render the result.
- The *View* with the help of the model data will render the result back to the user.

To understand the Spring MVC Framework we will now create a simple hello world example using the Eclipse IDE. I am using Exclipse IDE 3.4 , Spring IDE plugin, Tomcat 6.0 and Spring 3.0 to demonstrate this example.

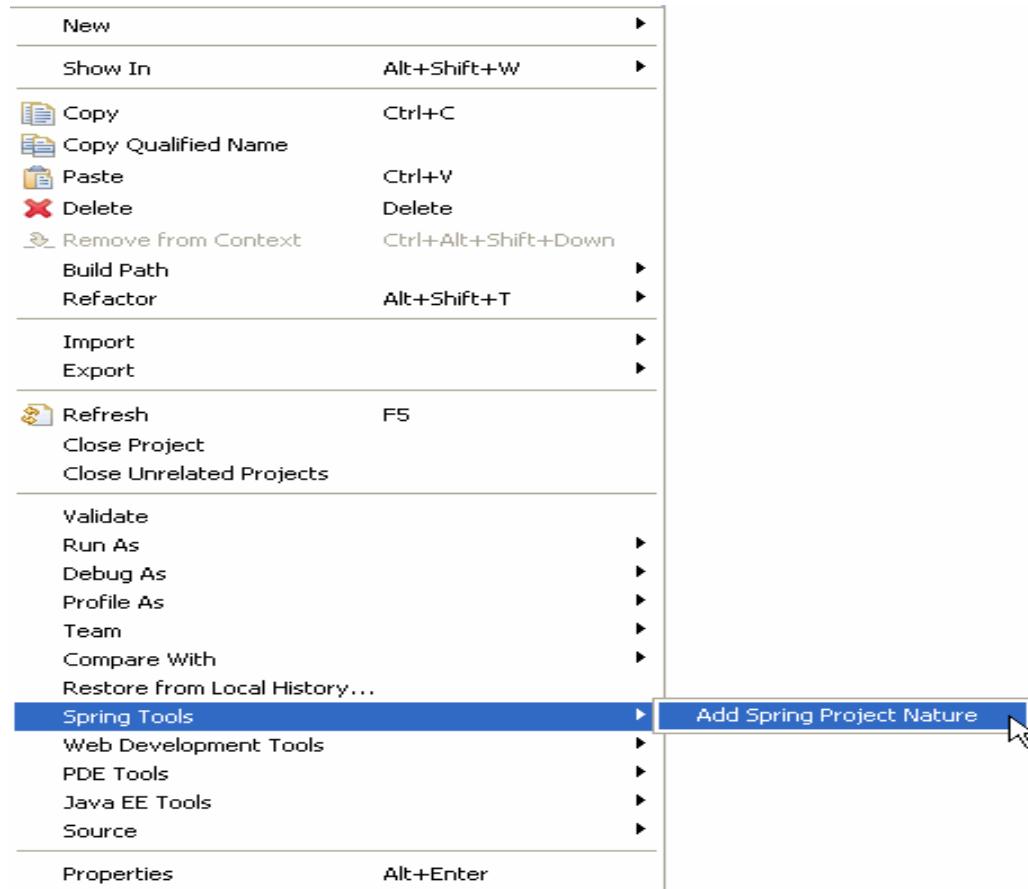
Go to *File -> New -> Dynamic Web Project*, to create a web project.



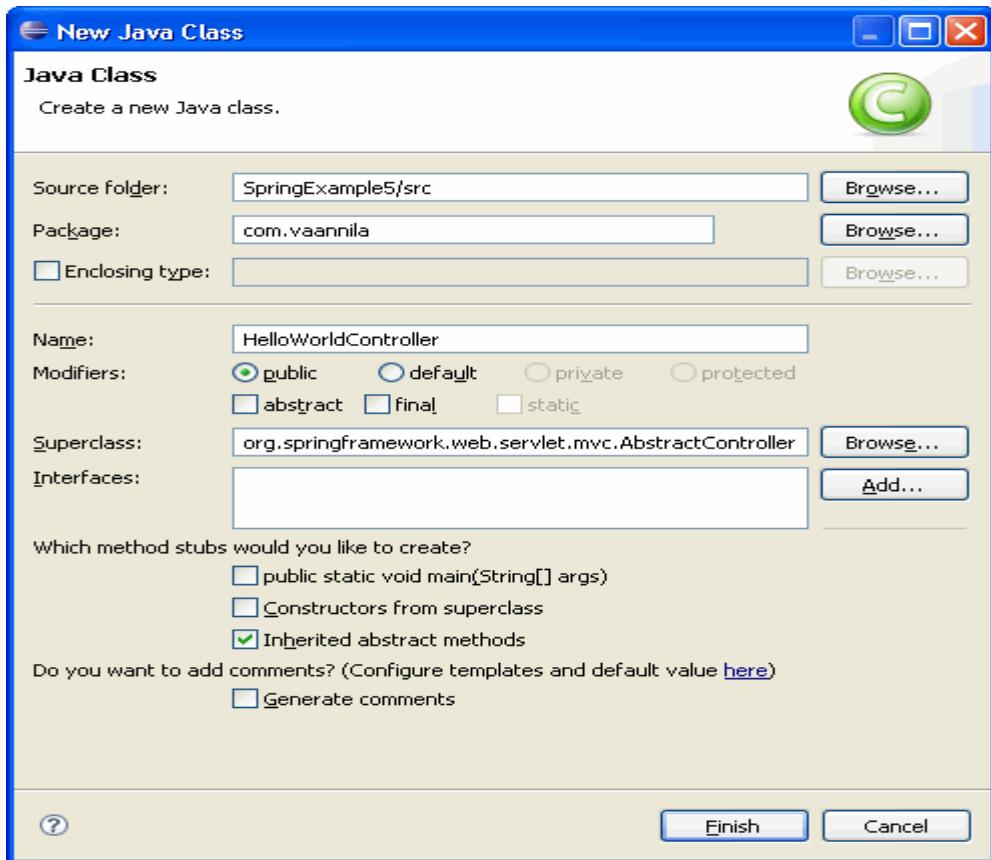
Enter the project name and click the *Finish* button.



Right click the project folder, and select *Spring Tools* -> *Add Spring Project Nature*, to add Spring capabilities to the web project. This feature will be available once you install the Spring IDE.



Create a new package `com.vaannila` inside the `src` directory. The Spring controller class extends `org.springframework.web.servlet.mvc.AbstractController` class. To create a new controller class right click the `src` directory and create a new java class, enter the controller class name and super class name and the *Finish* button.

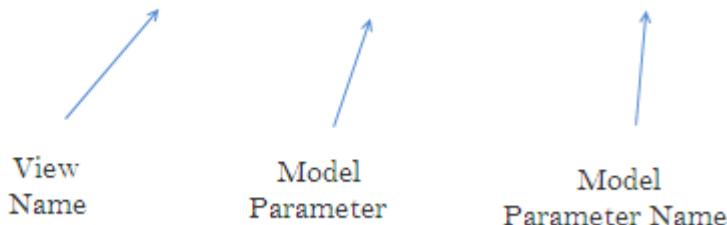


Copy the following code inside the *HelloWorldController* class.

```
view source
print?
01.import javax.servlet.http.HttpServletRequest;
02.import javax.servlet.http.HttpServletResponse;
03.
04.import org.springframework.web.servlet.ModelAndView;
05.import org.springframework.web.servlet.mvc.AbstractController;
06.
07.public class HelloWorldController extends AbstractController {
08.
09.private String message;
10.
11.@Override
12.protected ModelAndView handleRequestInternal(HttpServletRequest request,
HttpServletResponse response) throws Exception {
13.return new ModelAndView("welcomePage","welcomeMessage", message);
14.
15.
16.public void setMessage(String message) {
```

```
17. this.message = message;  
18.}  
19.  
20.}
```

```
return new ModelAndView("welcomePage","welcomeMessage", message);
```



The *DispatcherServlet*, as the name indicates, is a single servlet that manages the entire request-handling process. When a request is sent to the *DispatcherServlet* it delegates the job by invoking the appropriate controllers to process the request. Like any other servlet the *DispatcherServlet* need to be configured in the web deployment descriptor as shown.

[view source](#)

[print?](#)

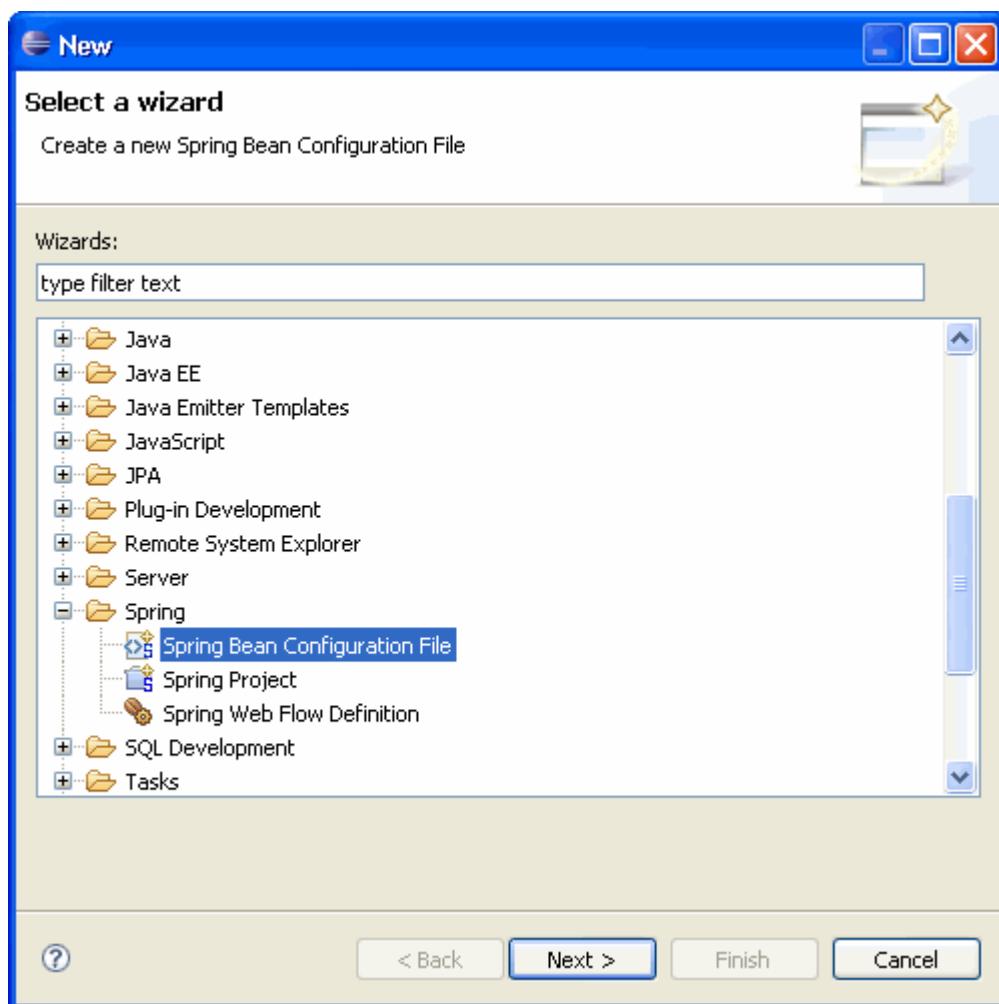
```
01.<?xml version="1.0" encoding="UTF-8"?>  
02.<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-  
instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/  
xml/ns/javaee/web-  
app_2_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaeehttp://java.sun.c  
om/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">  
03.<servlet>  
04.<servlet-name>dispatcher</servlet-name>  
05.<servlet-class> org.springframework.web.servlet.DispatcherServlet </servlet-class>  
06.<load-on-startup>1</load-on-startup>  
07.</servlet>  
08.<servlet-mapping>  
09.<servlet-name>dispatcher</servlet-name>  
10.<url-pattern>*.htm</url-pattern>  
11.</servlet-mapping>  
12.<welcome-file-list>  
13.<welcome-file>redirect.jsp</welcome-file>  
14.</welcome-file-list>
```

15.</web-app>

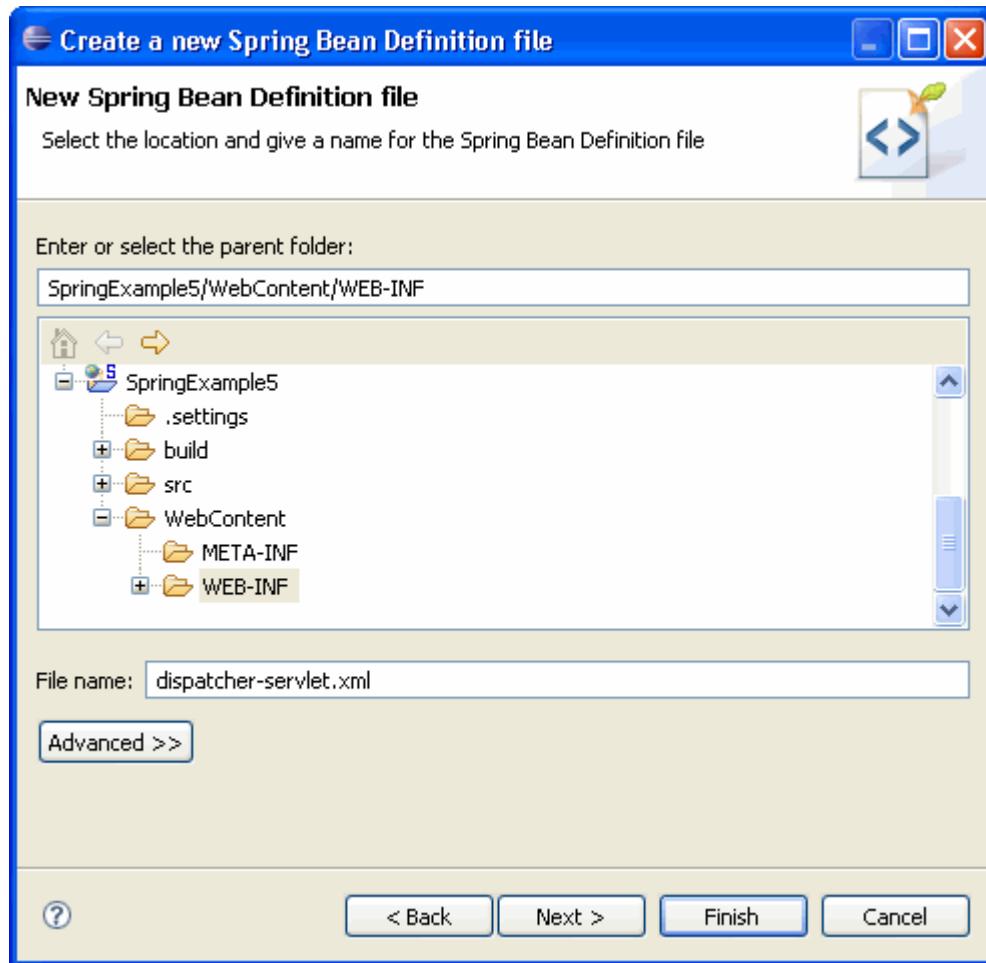
Here the servlet name is *dispatcher*. By default the *DispatcherServlet* will look for a file name *dispatcher-servlet.xml* to load the Spring MVC configuration. This file name is formed by concatenating the servlet name ("dispatcher") with "-servlet.xml". Here we user the the *url-pattern* as ".htm" inorder to hide the implementations technology to the users.

The *redirect.jsp* will be invoked first when we execute the Spring web application. This is the only *jspfile* outside the *WEB-INF* directory and it is here to provide a redirect to the *DispatcherServlet*. All the other views should be stored under the *WEB-INF* directory so that they can be invoked only through the controller process.

To create a bean configuration file right click the *WebContent* folder and select *New -> Other*. The following dialog box appears.



Select the Spring Bean Configuration file and click Next.



Enter the file name as "*dispatcher-servlet.xml*" and click the *Finish* button.

Now the Spring bean configuration file is created, we need to configure the *Controller* and the *ViewResolver* classes. The following code shows how to do this.

[view source](#)

[print?](#)

```
01.<bean id="viewResolver"
02.  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
03.  <property name="prefix">
04.    <value>/WEB-INF/jsp/</value>
05.  </property>
06.  <property name="suffix">
07.    <value>.jsp</value>
08.  </property>
09.</bean>
10.
11.<bean name="/welcome.htm" class="com.vaannila.HelloWorldController" >
12.  <property name="message" value="Hello World!" />
13.</bean>
```

```
14.  
15.</beans>
```

First let's understand how to configure the controller.

```
1.<bean name="/welcome.htm" class="com.vaannila.HelloWorldController" >  
2.<property name="message" value="Hello World!" />  
3.</bean>
```

Here the **name** attribute of the *bean* element indicates the **URL pattern to map** the request. Since the *id* attribute can't contain special characters like "/", we specify the URL pattern using the *name* attribute of the *bean* element. By default the *DispatcherServlet* uses the *BeanNameUrlHandlerMapping* to map the incoming request. The *BeanNameUrlHandlerMapping* uses the bean name as the URL pattern. Since *BeanNameUrlHandlerMapping* is used by default, you need not do any separate configuration for this.

We set the message attribute of the *HelloWorldController* class thru setter injection. The *HelloWorldController* class is configured just like an another JavaBean class in the Spring application context, so like any other JavaBean we can set values to it through Dependency Injection(DI).

The *redirect.jsp* will redirect the request to the *DispatcherServlet*, which inturn consults with the *BeanNameUrlHandlerMapping* and invokes the *HelloWorldController*.

The *handleRequestInternal()* method in the *HelloWorldController* class will be invoked. Here we return the *message* property under the name *welcomeMessage* and the view name *welcomePage* to the *DispatcherServlet*. As of now we only know the view name, and to find the actual view to invoke we need a *ViewResolver*.

The *ViewResolver* is configured using the following code.

[view source](#)

print?

```
01.<bean id="viewResolver"  
02. class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
03.<property name="prefix">  
04.<value>/WEB-INF/jsp/</value>  
05.</property>  
06.<property name="suffix">  
07.<value>.jsp</value>  
08.</property>  
09.</bean>
```

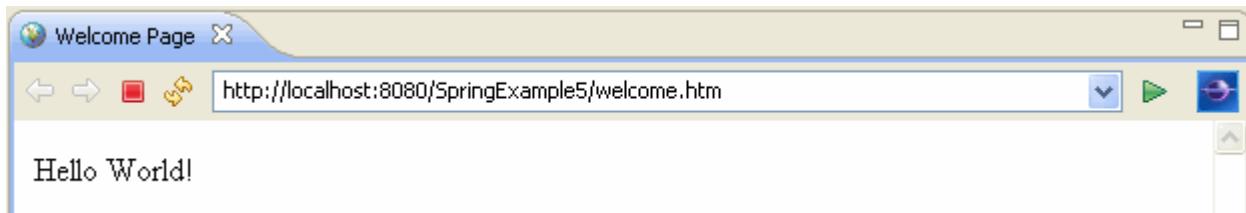
Here the *InternalResourceViewResolver* is used to resolve the view name to the actual view. The *prefix value + view name + suffix value* will give the actual view location. Here the actual view location is */WEB-INF/jsp/welcomePage.jsp*

The following library files are needed to run the example.

[view source](#)
[print?](#)

01.antlr-runtime-3.0
02.common-logging-1.0.4
03.org.springframework.asm-3.0.0.M3
04.org.springframework.beans-3.0.0.M3
05.org.springframework.context-3.0.0.M3
06.org.springframework.context.support-3.0.0.M3
07.org.springframework.core-3.0.0.M3
08.org.springframework.expression-3.0.0.M3
09.org.springframework.web-3.0.0.M3
10.org.springframework.web.servlet-3.0.0.M3

To execute the example run the *redirect.jsp* file. The following page will be displayed.



1) Introduction

The **Spring MVC** provides rich functionality for building robust **Web Applications** and it is available as a separate module in the Distribution. As a pre-requisite, readers are advised to go through the introductory article on **Spring Framework Introduction to Spring Framework**. The **Spring MVC Framework** is architected and designed in such a way that every piece of logic and functionality is highly configurable. Also Spring can integrate effortlessly with other popular Web Frameworks like **Struts**, **WebWork**, **Java Server Faces** and **Tapestry**. It means that you can even instruct Spring to use any one of the Web Frameworks. More than that Spring is not tightly coupled with Servlets or Jsp to render the View to the Clients. Integration with other View technologies like **Velocity**, **Freemarker**, **Excel** or **Pdf** is also possible now. This article provides an introduction over the various components that are available in the **Spring MVC** for the Web Tier. Specifically the major Core Components like **Dispatcher Servlet**, **Handler Mappings**, **Controller**, **Model**, **View** and **View Resolver** along with the appropriate API are discussed briefly. Finally the article will conclude by presenting a Sample Application.

also read:

- [Spring Interview Questions](#)
- [Spring Framework Books \(recommended\)](#)
- [JSF Interview Questions](#)
- [Introduction to Spring MVC](#)

Spring

2) The Spring Workflow

Before taking a look over the various Components that are involved in the **Spring MVC Framework**, let us have a look on the style of Spring Web Flow.

1. The Client requests for a **Resource** in the Web Application.
2. The **Spring Front Controller**, which is implemented as a Servlet, will intercept the Request and then will try to find out the appropriate **Handler Mappings**.
3. The **Handle Mappings** is used to map a request from the Client to its Controller object by browsing over the various Controllers defined in the Configuration file.
4. With the help of **Handler Adapters**, the Dispatcher Servlet will dispatch the Request to the Controller.
5. The Controller processes the Client Request and returns the **Model and the View** in the form of ModelAndView object back to the Front Controller.
6. The **Front Controller** then tries to resolve the actual **View** (which may be Jsp, Velocity or Free marker) by consulting the **View Resolver object**.
7. Then the selected View is rendered back to the Client.

Let us look into the various Core Components that make up the Spring Web Tier. Following are the components covered in the next subsequent sections.

3) Dispatcher Servlet

The **Dispatcher Servlet** as represented by org.springframework.web.servlet.DispatcherServlet, follows the **Front Controller Design Pattern** for handling Client Requests. It means that whatever Url comes from the Client, this Servlet will intercept the Client Request before passing the Request Object to the Controller. The **Web Configuration file** should be given definition in such a way that this Dispatcher Servlet should be invoked for Client Requests. Following is the definition given in the web.xml to invoke **Spring's Dispatcher Servlet**.

web.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.4">
3
4      <servlet>
5          <servlet-name>dispatcher</servlet-name>
6          <servlet-class>
7              org.springframework.web.servlet.DispatcherServlet
8          </servlet-class>
9          <load-on-startup>2</load-on-startup>
10
11     <servlet-mapping>
12         <servlet-name>dispatcher</servlet-name>
13         <url-pattern>*.*</url-pattern>
```

```
14    </servlet-mapping>
15
16  </web-app>
17
```

Look into the definition of servlet-mapping tag. It tells that whatever be the Client Request (represented by *.* meaning any Url with any extension), invoke the Servlet by name'dispatcher'. In our case, the 'dispatcher' servlet is nothing but an instance of type'org.springframework.web.servlet.DispatcherServlet'.

Closing associated term with the Dispatcher Servlet is the **Application Context**. An **Application Context** usually represents a set of **Configuration Files** that are used to provide Configuration Information to the Application. The Application Context is a Xml file that contain various **Bean Definitions**. By default the Dispatcher Servlet will try to look for a file by name <servlet-name>-servlet.xml in the WEB-INF directory. So, in our case the Servlet will look for a file name called dispatcher-servlet.xml file in the WEB-INF directory.

It is wise sometimes to split all the Configuration information across multiple Configuration Files. In such a case we have to depend on a **Listener Servlet** called **Context Loader** represented by org.springframework.web.context.ContextLoaderListener.

```
1  <web-app>
2
3  <listener>
4    <listener-class>
5      org.springframework.web.context.ContextLoaderListener
6    </listener-class>
7  </listener>
8
9  </web-app>
```

By default, this Context Listener will try to look for the Configuration File by name 'applicationContext.xml' in the '/WEB-INF' directory. But with the help of the parameter'contextConfigLocation' the default location can be overridden. Even multiple Configuration Files each containing separate piece of Information is also possible.

web.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.4">
3
4    <listener>
5      <listener-class>
6        org.springframework.web.context.ContextLoaderListener
7      </listener-class>
8    </listener>
9
10   <context-param>
11     <param-name>contextConfigLocation</param-name>
12     <param-value>/WEB-INF/contacts.xml, /WEB-INF/resources.xml</param-value>
13   </context-param>
14
15   </web-app>
```

The above definition instructs the Framework to look and load for the Configuration Files by name 'contacts.xml' and 'resources.xml' in the WEB-INF directory.

4) Handler Mappings

When the Client Request reaches the **Dispatcher Servlet**, the **Dispatcher Servlet** tries to find the appropriate **Handler Mapping** Object to map between the Request and the Handling Object. A **Handler Mapping** provides an abstract way that tell how the Client's Url has to be mapped to the Handlers. Four concrete variation of Handler Mapping are available. They are defined as follows

- BeanNameUrl HandlerMapping
- CommonsPathMap HandlerMapping
- ControllerClassName HandlerMapping
- SimpleUrl HandlerMapping

All the above Handler Mapping objects are represented

as BeanNameUrlHandlerMapping, CommonsPathMapHandlerMapping, ControllerClassNameHandlerMapping and SimpleUrlHandlerMapping in the org.springframework.web.servlet package respectively. Let us see the functionalities and the differences in usage one by one.

4.1) BeanNameUrl HandlerMapping

This is the simplest of the **Handler Mapping** and it is used to map the Url that comes from the Clients directly to the Bean Object. In the later section, we will see that the Bean is nothing but a Controller object. For example, consider that the following are the valid Url in a Web Application that a Client Application can request for.

- 1 http://myserver.com/eMail/showAllMails
- 2
- 3 http://myserver.com/eMail/composeMail
- 4
- 5
- 6 http://myserver.com/eMail/deleteMail

Note that the Url (excluding the Application Context) in the above cases are 'showAllMails', 'composeMail' and 'deleteMail'. This means that the Framework will look for Bean Definitions with Identifiers 'showAllMails', 'composeMail' and 'deleteMail'. Consider the following Xml code snippet in the Configuration file,

```
1 <beans>
2
3     <bean name="/showAllMails.jsp"
4         class="com.javabeat.net.ShowAllMailsController">
5     </bean>
6
7     <bean name="/composeMail.jsp"
8         class="com.javabeat.net.ComposeMailController">
9     </bean>
10
11    <bean name="/ deleteMail.jsp"
12        class="com.javabeat.net.DeleteMailController">
13    </bean>
```

```
13
14    </beans>
15
```

So, in **BeanNameUrl Handler Mapping**, the Url of the Client is directly mapped to the Controller. To enable this kind of Handler Mapping in the Application, the Configuration file should have a similar kind of definition like the following,

```
1   <beans>
2
3   ...
4   <bean id="beanNameUrl"
5     class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/>
6   ...
7
8   </beans>
```

4.2) CommonsPathMap HandlerMapping

This is a rarely used Handler Mapping in which case, the name of the Url to which the Controller has to be mapped is specified directly in the Source file of the Controller. Considering the previous example, if we want to map 'showAllMails', 'composeMail' and 'deleteMail' to Controllers namely ShowAllMailsController, ComposeMailController and DeleteMailController, then the mapping information must be specified in the form of **meta-data in the source files** inside the Javadoc comments. Consider the following Controller Definitions,

```
1
2  /**
3   * @@@ org.springframework.web.servlet.handler.commonsattributes.
4   * @PathMap("/showAllMails.jsp")
5   */
6  public class ShowAllMailsController{
7
8
8  /**
9   * @@@ org.springframework.web.servlet.handler.commonsattributes.
10  * @PathMap("/composeMail.jsp")
11  */
12  public class ComposeMailController{
13
14
14  /**
15   * @@@ org.springframework.web.servlet.handler.commonsattributes.
16   * @PathMap("/deleteMail.jsp")
17  */
18  public class DeleteMailController {
19
20}
```

The attribute must point to `org.springframework.web.servlet.handler.commonsattributes.PathMap`. By defining Controllers in this way, one more additional compilation step is needed. That is to make the availability of this

attribute in the Class files, this [Java](#) Source has to be compiled with the **Commons Attribute Compiler** which comes along with the Spring Distribution. As before, to enable this kind of mapping , the Configuration File should have an entry similar to this,

```
1 <beans>
2
3   <bean id="metaHandlerMapping" class="org.springframework.web.servlet.handler.
4     metadata.CommonsPathMapHandlerMapping"/>
5
6 </beans>
7
```

4.3) ControllerClassName HandlerMapping

In this kind of Handler Mapping, the name of the Controller is taking directly from the Url itself with slight modifications. For example, let us assume that the Client request ends with Url as shown below,

```
1 http://myserver.com/emailApp/showInbox.jsp
2
3 http://myserver.com/emailApp/showDeletedItems.jsp
```

And as such, we have a Controller definition by name ShowController as follows,

ShowController.java

```
1 public class ShowController{
2 }
```

Also the Configuration file is made to activate this kind of Handler Mapping by making the following definition,

```
1 <beans>
2
3   <bean id="controllerClassName" class="org.springframework.web.servlet.handler.
4     metadata.ControllerClassNameHandlerMapping"/>
5
6 </beans>
```

The first thing the Framework does it, it will traverse through the List of Controllers defined in the Configuration File and perform these actions. For the Controller ShowController, then Framework will remove the Controller String and then lowercase the first letter. In our case the string now becomes show. Now whatever Client Request matches the pattern /show*, then theShowController will be invoked.

4.4) SimpleUrl HandlerMapping

This is the Simplest of all the Handler Mappings as it directly maps the Client Request to some Controller object.

Consider the following Configuration File,

```
1 <bean id="simpleUrlMapping"
2   class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
3
```

```

4   <property name="mappings">
5     <props>
6       <prop key="/showAllMails.jsp">showController</prop>
7       <prop key="/composeMail.jsp">composeController</prop>
8       <prop key="/deleteMail.jsp">deleteController</prop>
9     </props>
10    </property>
11  </bean>
12

```

The set of mappings is encapsulated in the 'property' tag with each defined in a 'prop' element with the 'key' attribute being the Url, the value being the Identifier of the Controller Objects. Note that the Beans for the above Identifiers should be defined somewhere in the Configuration File.

5) Handler Adapters

It is important to understand that the Spring Framework is so flexible enough to define what Components should be delegated the Request once the **Dispatcher Servlet** finds the appropriate **Handler Mapping**. This is achieved in the form of **Handler Adapters**. If you remember in the Spring Work flow section, that it is mentioned once the Dispatcher Servlet chooses the appropriate **Handler Mapping**, the Request is then forwarded to the Controller object that is defined in the Configuration File. This is the default case. And this so happens because the **Default Handler Adapter** is **Simple Controller Handler Adapter** (represented

by org.springframework.web.servlet.SimpleControllerHandlerAdapter), which will do the job of the Forwarding the Request from the Dispatcher to the Controller object.

Other types of **Handler Adapters** are **Throwaway Controller**

HandlerAdapter(org.springframework.web.servlet.ThrowawayControllerHandlerAdapter) and **SimpleServlet HandlerAdapter**(org.springframework.web.servlet.SimpleServletHandlerAdapter). The **Throwaway Controller HandlerAdapter**, for example, carries the Request from the Dispatcher Servlet to the **Throwaway Controller** (discussed later in the section on Controllers) and Simple Servlet Handler Adapter will carry forward the Request from the Dispatcher Servlet to a Servlet thereby making the Servlet.service() method to be invoked.

If, for example, you don't want the default **Simple Controller Handler Adapter**, then you have to redefine the Configuration file with the similar kind of information as shown below,

```

1   <bean id="throwawayHandler" class = "org.springframework.web.servlet.mvc.throwaway.
2     ThrowawayControllerHandlerAdapter"/>
3
4   or
5
6   <bean id="throwawayHandler" class="org.springframework.web.servlet.mvc.throwaway.
7     SimpleServletHandlerAdapter"/>
8

```

Even, it is possible to write a **Custom Handler Adapter** by implementing the HandlerAdapterinterface available in the org.springframework.web.servlet package.

6) Controller

Controllers are components that are being called by the Dispatcher Servlet for doing any kind of Business Logic. Spring Distribution already comes with a variety of **Controller Components** each doing a specific purpose. All Controller Components in Spring implement the org.springframework.web.servlet.mvc.Controller interface. This section aimed to provide the commonly used Controllers in the Spring Framework. The following are the Controller Components available in the Spring Distribution.

- SimpleFormController
- AbstractController
- AbstractCommandController
- CancellableFormController
- AbstractCommandController
- MultiActionController
- ParameterizableViewController
- ServletForwardingController
- ServletWrappingController
- UrlFilenameViewController

The following section covers only

on AbstractController, AbstractCommandController, SimpleFormController and CancellableFormController in detail.

6.1) Abstract Controller

If one wants to implement **Custom Controller Component** right from the scratch, then instead of implementing the Controller interface, extending AbstractController can be preferred as it provides the basic support for the **GET and the POST** methods. It is advised that only for simple purpose, this type of extensions should be used. The purpose may be as simple as returning a resource to the Client upon request without having the need to examine the Request Parameters or other Stuffs. For example, consider the following piece of code,

MySimpleController.java

```
1  public class MySimpleController extends AbstractController{  
2  
3      public void handleRequestInternal(HttpServletRequest request,  
4          HttpServletResponse response){  
5  
6          return new ModelAndView("myView");  
7  
8      }  
9  }
```

Note that the Dispatcher Servlet will call the handleRequest() method by passing the Request and the Response parameters. The implementation just returns a ModelAndView (discussed later) object with myView being the logical view name. There are Components called **View Resolvers** whose job is to provide a mapping between the **Logical View Name** and the actual **Physical Location of the View Resource**. For the time being, assume that somehow, myView is mapped to myView.jsp. So, whenever the Dispatcher Servlet invokes this MySimpleController object, finally myView.jsp will be rendered back to the Client.

6.2) Abstract Command Controller

The concept of **Command Controller** comes into picture when the Business Logic depends upon the values that are submitted by the User. Instead of depending on the Servlet Api to get the Request Parameter Values and other session Objects, we can depend on this **Abstract Command Controller** to take those pieces of Information. For example consider the following code snippet which has a simple business logic telling that, depending on the existence of username, display the form success.jsp or failure.jsp

MySimpleController.java

```
1  public class MySimpleController extends AbstractCommandController{  
2  
3      public MySimpleController(){  
4          setCommandClass(UserInfo.class);  
5      }  
6  
7      public void handle(HttpServletRequest request, HttpServletResponse response,  
8      Object command){  
9  
10         UserInfo userInfo = (UserInfo)command;  
11         if ( exists(userInfo.getUserName)){  
12             return new ModelAndView("success");  
13         }else{  
14             return new ModelAndView("failure");  
15         }  
16  
17         private boolean exists(String username){  
18             // Some logic here.  
19         }  
    }
```

Note that the **Client Parameters** (username , in this case) is encapsulated in a simple Class called UserInfo which is given below. The value given by the Client for the username field will be directly mapped to the property called username in the UserInfo. In the Constructor of the MySimpleController class, we have mentioned the name of the Command Class which is going to hold the Client Request Parameters by calling the setCommandClass() method. Also note that in the case of Command Controller, the method that will be called by the Dispatcher Servlet will be handle() which is passed with the Command object apart from the Request and the Response objects.

UserInfo.java

```
1  public class UserInfo{  
2  
3      private String username;  
4      // Getters and Setters here.  
5  
6  }
```

6.3) Simple Form Controller

Asking the User to fill in a Form containing various information and submitting the form normally happens in almost every Web Application. The **Simple Form Controller** is exactly used for that purpose. Let us give a simple example to illustrate this. Assume that upon Client Request a page called empInfo.jsp is rendered to the client containing empName, empAge and empSalaryfields. Upon successful completion a Jsp Page called empSuccess.jsp is displayed back to the Client. Now let us see how we can make use of the Simple Form Controller to achieve this kind functionality.

The very first thing is that, to collect the Client Input Values, a Command object which contains getter and setters must be defined. Following the skeleton of the class called EmplInfo.

EmplInfo.java

```
1  public class EmplInfo{  
2  
3      private String empName;  
4      private int empAge;  
5      private double empSalary;  
6  
7      // Getters and setters for the above properties.  
8  
9  }
```

The next thing is to write a class that extends SimpleFormController. But this time, the doSubmitAction() method should be overridden. This is the method that will be called when the Client submits the form. Following is the definition of the Controller class.

EmpFormController.java

```
1  public class EmpFormController extends SimpleFormController{  
2  
3      public EmpFormController(){  
4          setCommandClass(EmplInfo.class);  
5      }  
6  
7      public void doSubmitAction(Object command){  
8          EmplInfo info = (EmplInfo)command;  
9          process(info);  
10     }  
11  
12     private void process(EmplInfo info){  
13         //Do some processing with this object.  
14     }  
15 }
```

As we mentioned previously, the form that collects information from the Client is empInfo.jsp and upon successful submission the view empSuccess.jsp should be displayed. This information is externalized from the Controller class and it is maintained in the Configuration File like the following,

```
1  <bean id = "empForm" class="EmpFormController">  
2
```

```

3   <property name="formView">
4     <value>emplInfo</value>
5   </property>
6
7   <property name="successView">
8     <value>empSuccess</value>
9   </property>
10
11 </bean>

```

Note the two property names 'formView' and 'successView' along with the values 'emplInfo' and 'empSuccess'. These properties represent the initial View to be displayed and the final view (after successful Form submission) to be rendered to the Client.

6.4) Cancellable FormController

If you carefully notice with the implementation of Simple Form Controller, there are ways to provide the Initial and the Successful View to the Clients. But what happens when the Form is cancelled by the User? Who will process the Cancel operation of the Form?

The above issues can be given immediate solution with the usage of **Cancellable FormController**. The good thing is that Cancellable FormController extends **SimpleForm Controller** so that all the functionalities are visible to this Controller also. Suppose say that the User clicks the cancel button, the Framework will check in the Request parameter for a key with name 'cancelParamKey'. If it is so, then it will call the onCancel() method. Consider the following definition,

MyCompleteFormController.java

```

1  public class MyCompleteFormController extends CancellableFormController{
2
3    public ModelAndView onCancel(){
4      return new ModelAndView("cancelView");
5    }
6  }

```

7) Model And View

Model and View (represented by the class `org.springframework.web.servlet.ModelAndView`) is returned by the Controller object back to the Dispatcher Servlet. This class is just a Container class for holding the Model and the View information. The Mode object represents some piece of information that can be used by the View to display the information. Both these Objects are given high degree of abstraction in the Spring Framework.

Any kind of **View Technology** (`org.springframework.web.servlet.View`) can be plugged into the Framework with ease. For example, Excel, Jasper Reports, Pdf, Xslt, Free Marker, Html, Tiles, Velocity etc. are the supported Frameworks as of now. The Model object (represented by `org.springframework.ui.ModelMap`) is internally maintained as a Map for storing the Information.

Following are the ways to Construct the Model and the View object.

```
1  View pdfView = ...;
```

```

2     Map modelData = new HashMap();
3
4     ModelAndView mv1 = new ModelAndView(pdfView, modelData);

```

The above constructs a ModelAndView object by passing the actual View object along with the Model object. Now consider the following code,

```
1     ModelAndView mv1 = new ModelAndView("myView", someData);
```

Note, in the above example, a string with "myView" is passed for the View. This way of specifying a **View** is called a **Logical View**. It means that myView either can point to something called myView.jsp or myView.pdf or myView.xml. The **Physical View Location** corresponding to the **Logical View** can be made configurable in the Configuration File.

8) View Resolver

In the previous section, we talked about **Logical View** and the **Physical View Location** for the Logical View. The mapping between the Logical name and the Physical View Location is taken care by the **View Resolver** object. Without any surprise, Spring comes with a set of **Built-In Spring Resolvers**. It is even possible to write **Custom View Resolvers** by implementing the org.springframework.web.servlet.ViewResolver interface. Following are the available View Resolvers in the Spring Distribution.

- BeanNameViewResolver
- FreeMarkerViewResolver
- InternalResourceViewResolver
- JasperReportsViewResolver
- ResourceBundleViewResolver
- UrlBasedViewResolver
- VelocityLayoutViewResolver
- VelocityViewResolver
- XmlViewResolver
- XsltViewResolver

The following section concentrates only on **Internal Resource View Resolver** and **Bean Name View Resolver** in detail.

8.1) Internal Resource View Resolver

The **Internal Resource View Resolver** will try to map the Logical name of the Resource as returned by the Controller object in the form of ModelAndView object to the **Physical View location**. For example, consider the following class definition which returns different ModelAndView objects.

MyController.java

```

1     public class MyController {
2
3         public void handle(){
4             if(condition1()){
5                 return new ModelAndView("myView1");
6             }else if (condition2()){
7                 return new ModelAndView("myView2");
8             }
9             return new ModelAndView("myView3");

```

```
9      }
10     }
11
```

Assume that if the Client Request satisfies condition1(), then the view myView.jsp which is present in the /WEB-INF folder should be displayed and for the client Requests satisfying condition2() and the other one, myView2.jsp and myView3.jsp should be displayed.

For this to happen, the following entry must be made in the Configuration File,

```
1 <bean id="viewResolver" class="org.springframework.web.servlet.view.
2 InternalResourceViewResolver">
3
4   <property name="prefix"><value>/WEB-INF/</value></property>
5   <property name="suffix"><value>.jsp</value></property>
6
7 </bean>
```

This is how the **Internal Resource View Resolver** will map the Logical View Name to the physical Location. When the logical View name is myView1, then it will construct a view name which is the summation of **the prefix + the logical View Name + the suffix**, which is going to be /WEB-INF/myView.jsp. The same is the case for myView2.jsp and myView3.jsp.

8.2) Bean Name View Resolver

One of the dis-advantage of using **Internal Resource View Resolver** is that the name of the View file (whether it is a Jsp File or the Pdf File) must be present in the Web Application Context. Dynamically generated View files may not be possible. In such a case, we may use the **Bean Name View Resolver** which will dynamically generate View in Pdf or Excel Formats.

For the example, if the ModelAndView object represents a View by name “pdf” as shown in the following snippet,

```
1 return ModelAndView("pdf")
```

And, if we want to generate the Pdf file, then we should have defined the Configuration information in the file as follows,

```
1 <bean id="beanNameResolver"
2   class="org.springframework.web.servlet.view.BeanNameViewResolver"/>
```

The above code configures the Framework to use BeanNameViewResolver. Since the logical name ‘pdf’ must resolve to a Bean Name, we should define a similar entry like the following in the Configuration File. Note that, in the following MyPdfGenerator may be the sub-class

of org.springframework.web.servlet.view.document.AbstractPdfView for generating the Pdf File.

```
1 <bean id = " pdf " class = "MyPdfGenerator"/>
```

9) Sample Application

9.1) Introduction

The final Section of this article details a Simple Contact Application that has provisions for Creating, Deleting and Listing Contact Objects. The aim of this Application is to show the various use of Controller Components like Abstract Controller, Abstract Command Controller and Form Controller along with Configuration Information.

9.2) The Web Descriptor File

As mentioned previously, since the **Dispatcher Servlet** acts as an **Interceptor** for the Client Request, an entry for the same has to be mentioned in the web.xml file. Follow is the code snippet for the same,

web.xml

web.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
5
6  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
7
8  <servlet>
9    <servlet-name>dispatcher</servlet-name>
10   <servlet-class>
11     org.springframework.web.servlet.DispatcherServlet
12   </servlet-class>
13   <load-on-startup>2</load-on-startup>
14 </servlet>
15
16 <servlet-mapping>
17   <servlet-name>dispatcher</servlet-name>
18   <url-pattern>*.htm</url-pattern>
19 </servlet-mapping>
20 </web-app>
```

9.3) Configuration File

The following represents the Configuration File for holding various piece of Configuration Information. The first thing to note is the type of Handler Mapping configured. In our case, it is the Bean Name Url Handler Mapping which means that the Url of the Client is tightly coupled with the class name of the Bean (Controller). Since all the Jsp files are maintained in the '/WEB/contacts' directory the 'prefix' property is pointing to '/WEB/contacts'.

For the Create, Delete and List operation on Contacts, three different Controller Components have been defined. They are CreateContactController, DeleteContactController and ListContactsController respectively.

```

dispatcher-servlet.xml
    <?xml version="1.0" encoding="UTF-8" ?>
1     <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
2     "http://www.springframework.org/dtd/spring-beans.dtd">
3     <beans>
4
5         <bean id="beanNameUrlMapping" class="org.springframework.web.servlet.handler.
6             BeanNameUrlHandlerMapping"/>
7
8         <bean name = "/CreateContact.htm" class="net.javabeat.articles.spring.mvc.
9             contacts.CreateContactController">
10
11            <property name="formView">
12                <value>CreateContact</value>
13            </property>
14            <property name="successView">
15                <value>ContactCreated</value>
16            </property>
17
18        </bean>
19
20        <bean name = "/DeleteContact.htm" class= "net.javabeat.articles.spring.mvc.
21             contacts.DeleteContactController">
22        </bean>
23
24        <bean name = "/ListContacts.htm" class= "net.javabeat.articles.spring.mvc.
25             contacts.ListContactsController">
26        </bean>
27
28        <bean id="viewResolver" class="org.springframework.web.servlet.view.
29             InternalResourceViewResolver">
30            <property name="prefix" value="/WEB-INF/contacts/" />
31            <property name="suffix" value=".jsp" />
32        </bean>
33
34    </beans>

```

9.4) CreateContact and ContactCreated Jsp Files

The following is the code for CreateContact.jsp file.

CreateContact.jsp

```

1     <html>
2     <head>
3     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4
5         <title>Create a Contact</title>
6     </head>
7     <body>

```

```

8     <h1>Create a Contact</h1>
9
10    <form name = "CreateContact" method = "get">
11        <input type = "text" name = "firstname" />
12        <input type = "text" name = "lastname" />
13        <br>
14        <input type="submit" name = "Create Contact" value = "Create Contact"/>
15    </form>
16
17    </body>
18    </html>
19

```

Note that since this is the page that will be shown to the user initially, in the Configuration file, the property 'formView' is pointed to 'CreateContact'. Following is the code for ContactCreated.jsp. Since this is the View that will be shown after the Form Submission the property 'successView' is made to point to 'ContactCreated'.

ContactCreated.jsp

```

1     <html>
2         <head>
3             <meta http-equiv = "Content-Type" content = "text/html; charset = UTF-8">
4             <title>Contact is Created</title>
5         </head>
6
7         <body>
8
9             <h1>Contact is successfully Created</h1>
10
11        </body>
12    </html>

```

9.5) DeleteContact.jsp

Following is the complete listing for DeleteContact.jsp file. Note that this Jsp File is mapped toDeleteContactController in the Configuration File.

DeleteContact.jsp

```

1     <html>
2         <head>
3             <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4
5             <title>Delete Contact</title>
6         </head>
7         <body>
8
9             <h1>Delete Contact</h1>
10
11            <form name = "DeleteContact" method = "get">
12                <input type = "text" name = "firstname" />
13                <br>
14                <input type="submit" name = "DeleteContact" value = "Delete Contact"/>

```

```
14    </form>
15
16    </body>
17  </html>
18
```

9.6) ListContacts.jsp

This page is to list all the existing Contacts that were created before. It should be noted that the **Model Object** that holds all the Contact Information in the form of List is available in the ListContactsController. The **Model Information** from the Controller after getting bound to the **Request Scope** is being taken off from the View in the form of **Expression Language**.

Following is the listing for ListContacts.jsp

ListContacts.jsp

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4
5  <title>Showing All Contacts</title>
6  </head>
7  <body>
8
9  <h1>Showing All Contacts</h1>
10 <p> The following are the created Contacts </p>
11
12 <c:forEach items = "${allContacts}" var="contact">
13   <c:out value="${contact.firstname}" /><br>
14   <c:out value="${contact.lastname}" /><br>
15 </c:forEach>
16
17 </body>
18 </html>
```

9.7) Contacts.java

The following is the Class structure for Contacts.java for encapsulating the properties firstname and lastname.

Contact.java

```
1  package net.javabeat.articles.spring.mvc.contacts;
2
3  public class Contact {
4
5    private String firstName;
6    private String lastName;
7
```

```

8     public Contact() {
9
10    }
11
12    public Contact(String firstName, String lastName){
13        this.firstName = firstName;
14        this.lastName = lastName;
15    }
16
17    public String getFirstName() {
18        return firstName;
19    }
20
21    public void setFirstName(String firstName) {
22        this.firstName = firstName;
23    }
24
25    public String getLastName() {
26        return lastName;
27    }
28
29    public void setLastName(String lastName) {
30        this.lastName = lastName;
31    }
32
33
34    public int hashCode(){
35        return firstName.hashCode() + lastName.hashCode();
36    }
37
38
39    public boolean equals(Object object){
40        if (object instanceof Contact){
41            Contact second = (Contact)object;
42            return (firstName.equals(second.getFirstName()) &&
43                lastName.equals(second.getLastName()));
44        }
45        return false;
46    }
47
48    public String toString(){
49        return "[First Name = " + firstName + ", Last Name = " + lastName + "]";
50    }
51
52}

```

9.8) ContactService.java

This simple service class provides functionalities for creating, deleting and listing the Contact information. All the Controller Components makes use of this class to achieve their respective functionalities.

ContactService.java

```

1     package net.javabeat.articles.spring.mvc.contacts;
2

```

```

3 import java.util.*;
4
5 public class ContactService {
6
7     private static Map contacts = new HashMap();
8
9     public ContactService() {
10    }
11
12     public static Contact createContact(Contact contact){
13         contacts.put(new Integer(contact.hashCode()), contact);
14         return contact;
15     }
16
17     public static Contact createContact(String firstName, String lastName){
18         return createContact(new Contact(firstName, lastName));
19     }
20
21     public static boolean deleteContact(String firstName){
22         Iterator iterator = contacts.entrySet().iterator();
23         while (iterator.hasNext()){
24             Map.Entry entry = (Map.Entry)iterator.next();
25             Contact contact = (Contact)entry.getValue();
26             if (contact.getFirstName().equals(firstName)){
27                 contacts.remove(new Integer(contact.hashCode()));
28                 return true;
29             }
30         }
31
32         public static List listContacts(){
33             return toList(contacts);
34         }
35
36         private static List toList(Map contacts){
37             List contactList = new ArrayList();
38             Iterator iterator = contacts.entrySet().iterator();
39             while (iterator.hasNext()){
40                 Map.Entry entry = (Map.Entry)iterator.next();
41                 Contact contact = (Contact)entry.getValue();
42                 contactList.add(contact);
43             }
44             return contactList;
45         }
46     }

```

9.9) Controller Classes

Following is the listing for CreateContact Controller. Note that since the Model Information for creating a contact (for which the Client supplies the firstname and the lastname parameters) is the Contact class, call has been made in the Constructor to setCommandClass() by passing the class name of the Contact class.

CreateContactController.java

```
1 package net.javabeat.articles.spring.mvc.contacts;
2
3 import org.springframework.web.servlet.mvc.SimpleFormController;
4
5 public class CreateContactController extends SimpleFormController{
6
7     public CreateContactController() {
8         setCommandClass(Contact.class);
9     }
10
11     public void doSubmitAction(Object command){
12         Contact contact = (Contact)command;
13         ContactService.createContact(contact);
14     }
15 }
```

Note that the method doSubmitAction() doesn't return anything because the next Logical View to be displayed will be taken from the Configuration file which is represented by the property called 'successView'.

Following two classes are the Controller Components for Deleting and Listing Contacts. Note that in the case of Delete Operation, a Jsp Page (DeletedContact.jsp) containing information telling that the Contact has been Deleted will be displayed. But since for the Contact Listing operation, the model information containing a Collection of Contact Objects has to be passed from the Controller to the View and the same is achieved in the 3 argument constructor to ModelAndView.

DeleteContactController.java

```
1 package net.javabeat.articles.spring.mvc.contacts;
2
3 import javax.servlet.http.*;
4 import org.springframework.validation.BindException;
5 import org.springframework.web.servlet.ModelAndView;
6 import org.springframework.web.servlet.mvc.AbstractCommandController;
7
8 public class DeleteContactController extends AbstractCommandController{
9
10    public DeleteContactController(){
11        setCommandClass(Contact.class);
12    }
13
14    public ModelAndView handle(HttpServletRequest request,
15        HttpServletResponse response, Object command, BindException errors)
16        throws Exception {
```

```

17     Contact contact = (Contact)command;
18     ContactService.deleteContact(contact.getFirstName());
19     return new ModelAndView("DeletedContact");
20
21 }
22 }
23

```

Here is the listing for ListContactsController.java.

ListContactsController.java

```

package net.javabeat.articles.spring.mvc.contacts;

1 import java.util.List;
2 import javax.servlet.http.*;
3 import org.springframework.web.servlet.ModelAndView;
4 import org.springframework.web.servlet.mvc.AbstractController;
5
6 public class ListContactsController extends AbstractController{
7
8     public ListContactsController() {
9         }
10
11    public ModelAndView handleRequestInternal(HttpServletRequest request,
12        HttpServletResponse response) throws Exception{
13
14        List allContacts = ContactService.listContacts();
15        return new ModelAndView("ListContacts", "allContacts", allContacts);
16    }
17 }

```

*****Struts*****

Struts Framework is used to develop the web based application. It simplifies the development of web based applications.

By using Struts we can develop only web based applications. We can't develop the standard alone applications.

We can use any of the following frameworks to develop the web based applications.

The y are :

2. Struts 2.Spring 3.Jsf 4.Web work 5.OpenSymphony

Struts framework is a piece of software which **contains the solutions to commonly repeated problems** across multiple projects.

The following are the advantages of Frameworks like Struts and spring:-.

- Frameworks resolve the problems of **identifying the architecture**. Every framework like Struts and Spring is delivered with MVC2 architecture.
- If we use Servlets and Jsp's we have to develop our own Controllers. If we use frameworks like Struts and Spring internally they come with Controllers
Eg: **ActionServlet** in Struts
DispatcherServlet in Spring
- When we use any framework we no need to use RequestDispatcher code as well as we no need to hard code the resource path names. By using these frameworks we can configure them in the configuration files.
- If we use JDBC, Servlets, Jsp's to develop the form based applications we have to write huge amount of code to take care of Server side validations and displaying errors in the same form.
- By using JDBC, Servlets, Jsp's we have to provide huge amount of code to develop the I18n applications. (The programs which displays the output based on client regional Languages)
- The frameworks like Struts and spring delivered with set of predefined tag libraries. If we use Servlets and Jsp's we have to develop our own tag library which is difficult.
- When we use the frameworks like Struts and spring we can use pdf/velocity/jsf as view components.

Most of the experienced guys are develops the new frameworks.

Every Company uses their own frameworks to develop the projects .all these frameworks internally uses the other frameworks.

Infosys is using **Pinnacle** framework.

Apache Company has released Struts framework to develop web based applications based on Model2 Architecture.

Apache guys has released Struts framework in 2 versions.

They are: 1)Struts 1.x
 2)Struts 2.x

We can download the framework from in multiple flavors.i.e. we can download **only the Struts jar** file or we can download only documentation or example applications.We can download struts from following link<http://struts.apache.org/>

When we download the Struts software majorly we get the following 3 files.

1. Set of Struts related **jar files** (struts-core.jar, struts-taglib.jar).
2. The documentation is available in **Docs** folder.

3. All the sample examples are available in **apps** folder.

Developing First Struts based application:

We can create the Struts based application by copying the jar files manually and we can configure them in web.xml.

Instead of doing manual work we can develop our first struts based application from **struts-blank.war**.

- Create a folder whose folder name is project name. and copy the struts-blank.war file into above created folder and extract the contents of it by using following command

Jar -xvf struts-blank.war

When we download struts framework we get the following four components :-

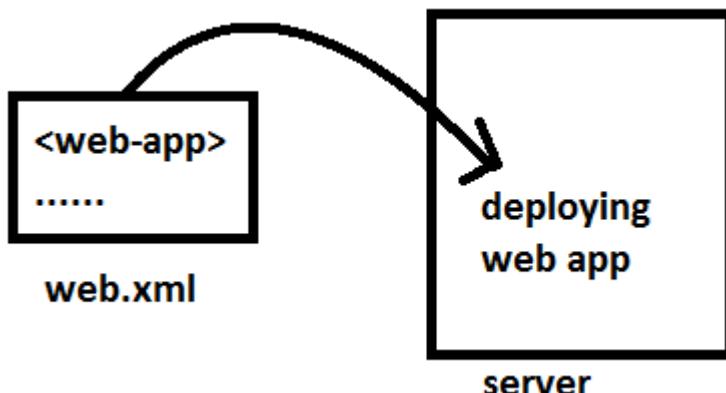
1. A servlet whose name is **org.apache.struts.action.ActionServlet**.
2. Set of predefined classes , some of them are :
Action, ActionMapping, ActionForward, ActionErrors, ActionError, ActionMessage, ActionForm.
3. Set of Struts tag libraries
 - a. Html
 - b. Bean
 - c. Logic
 - d. Nested
4. Get the **struts-config.xml** and **validation.xml** .

The following is the web.xml configuration :

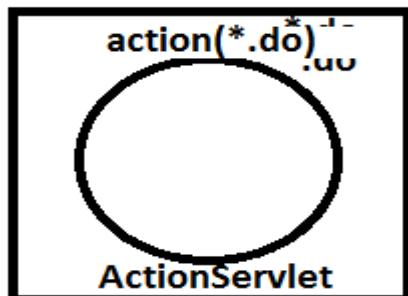
```
<web-app>
    <display-name>Struts Blank Application</display-name>
    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>
            org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
            <param-value>/WEB-INF/struts-config.xml
        </param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet-mapping>
```

The following steps will be carried out when we deploy Struts based project :

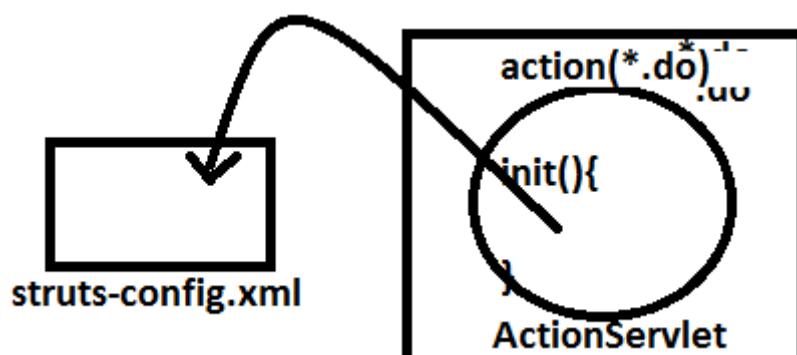
1. Server will read the contents from **web.xml** file and store the information in jvm's memory.



2. As there is **<load-on-startup>** tag for ActionServlet server has created ActionServlet object.



3. After server has created ActionServlet object server executed 1st `init(ServletConfig)` method and it will call the 2nd `init()` method. As part of `init()` method it will find the configuration file name details and by using parser program it read the contents from configuration file and store it in jvms memory.



The following is sample code available in predefined class ActionServlet

```

public class ActionServlet extends HttpServlet {
    public void init(){
        //get the file name
        String configFileName = getServletConfig().getInitParameter("config");
        //code to read the contents from configuration file and store that in jvms memory
    }
    public void service(HttpServletRequest request, HttpServletResponse response) {
    }
    public void destroy(){
        //code to remove the contents from jvms memory
    }
}

```

Whenever any client sends any request to server whose url pattern end with .doActionServlet service() throws an exception saying InvalidPathException: No action config found for the specified url.

A framework defines a procedure to carry out a task/ work.

We will get the path of the resource from url when remove the .do from path

/one.do
/any/thing/do
/student/login/do

Url's

/one
/any/thing
/student/login

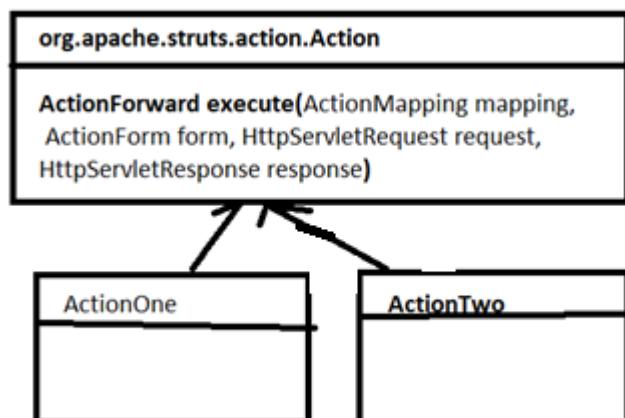
path of resources

In struts if we would like to carry out any work we need to develop Action classes.

Without Action class we can't carry out any work.

What is an Action class?

ans) A class which is a subclass of org.apache.struts.action.Action is called as Action class.



Developing Action classes:

The following is Action class

```

public class ActionOne extends Action {

    public ActionOne(){
        System.out.println("ActionOne object is created");
    }
    public ActionForward execute(ActionMapping mapping , ActionForm form ,
        HttpServletRequest request , HttpServletResponse response)throws IOException {
        System.out.println("we r in execute of ActionOne");
        ActionForward af = null;
        return af;
    }
}

```

ex:

To compile the above program we have to set the CLASSPATH to servlet-api.jar and struts-core.jar.

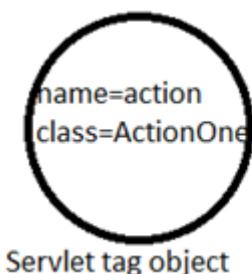
We must configure every Action class in struts-config.xml file.

```

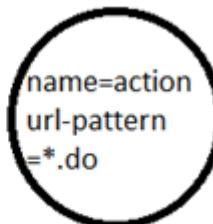
<struts-config>
    <action-mappings>
        <action type="ActionOne" path="/ao" />
    </action-mappings>
</struts-config>

```

Whenever we deploy a project in server it reads the content from web.xml file. Server represents the every tag in the form of Object. When we deploy the Struts project server has created the following two objects.

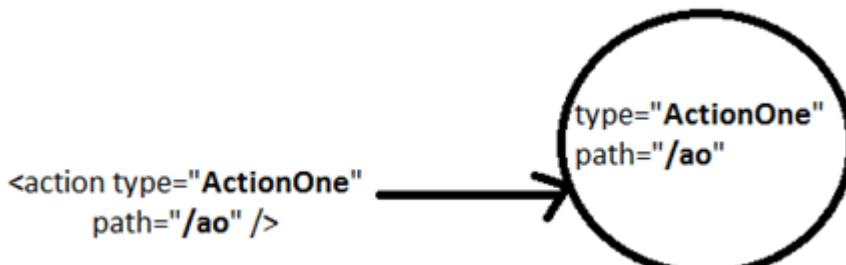


Servlet tag object



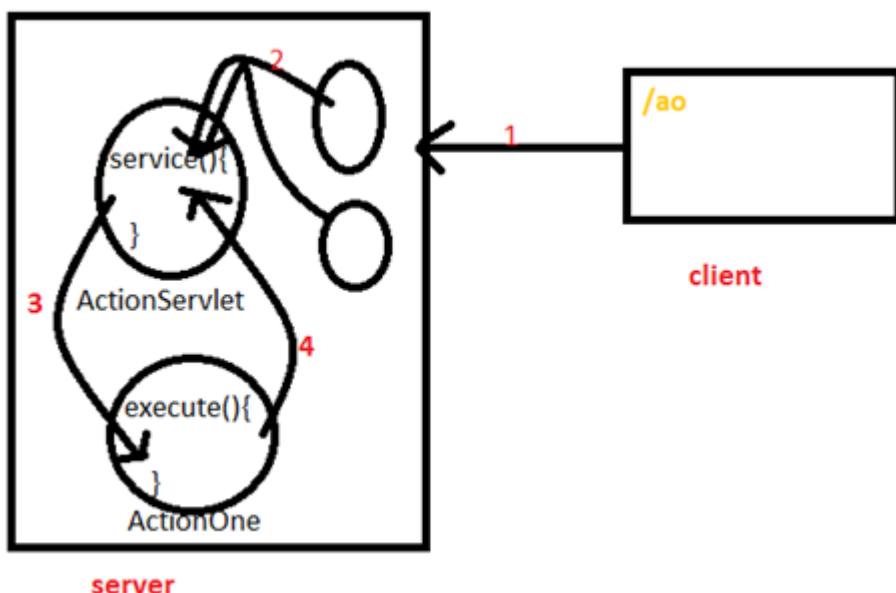
Servlet-mapping tag object

Whenever server has encounter an <action> tag in struts configuration file. It is creating an ActionMapping object .and it's containing the the path and type as shown below.



The following steps will be carried out when a client has send request to server whose url pattern end with .do.

1. Server creates request and response objects and handover to ActionServlet service().
2. ActionServlet service() get's the servletPath information using request object as shown below
request.getServletPath() Then it will remove the .do from that. Then it will get Path of the resource of Action class.
3. service() method checks is there any ActionMapping object is available whose path is matches to /ao.
4. If there is no ActionMapping object is available whose url pattern matches to /ao then server throw an exception **InvalidPathException**.
5. If ActionMapping object is available server gets the name of the Action class. then it will check is there any ActionOne object is available or not. If available it will not create any new ActionOne object other wise it will create a ActionOne class object using **Class.forName(ActionOne)**. So always we have to main default constructor in Action classes. Then ActionServlet service () will call the execute() using ActionOne object.
6. As ActionOne class execute() method returning ActionForward object with null value.
7. Whenever ActionServlet service() method has encountered a null value as part of ActionForward object. It stops execution of request. It will display a blank screen to the user.



To get the values from ActionMapping object for that we can use `getXxx()` methods of ActionMapping class as follows

```
public class ActionMappingDemo extends Action {  
    public ActionForward execute(ActionMapping mapping,  
        ActionForm form,HttpServletRequest request,  
        HttpServletResponse response ) throws IOException {  
        PrintWriter out = response.getWriter();  
        out.println("type is :" + mapping.getType() );  
        out.println("Path is " + mapping.getPath() );  
        out.println("Parameter is " + mapping.getParameter() );  
        return null;  
    }  
}
```

Whenever we change struts configuration file we must restart the server. This is because ActionServlet reads the contents from configuration file at the time deployment only.

We use parameter attribute in configuration file to supply an extra information to that particular action.

```
<action path="/ac" type="ActionMappingDemo" parameter="one" />
```

name attribute is used as logical name to identify a particular action. This name attributed used mainly in developing form based application.

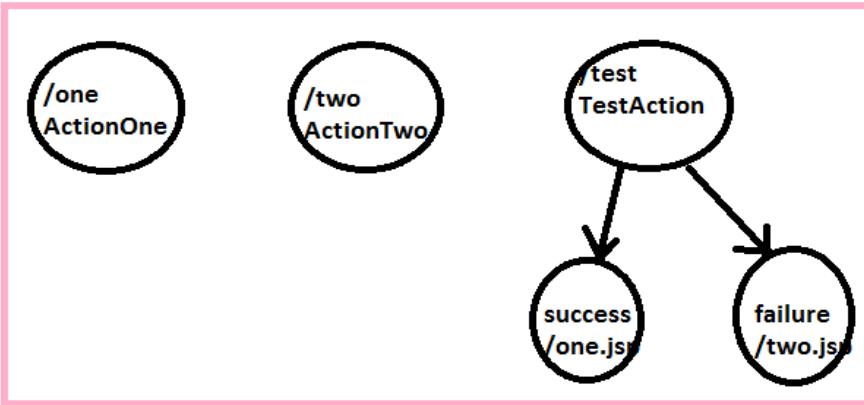
As part of the Struts configuration file added an extra tag as **<forward>**

```
<action path="/test" type="blmsr.jee.struts.TestAction" >  
    <forward name="success" path="/pages/one.jsp" />  
    <forward name="failure" path="/pages/error.jsp" />  
</action>
```

struts-config.xml

when we deploy the Project for the above configuration server reads the web.xml and then creates the ActionServlet object and call the init() .

As part of this server reads the configuration file and when Parser encounters the **<action>** tag and **<forward>** then it will creates ActionMapping and ActionForward objects as shown below



To check whether an ActionMapping object contains a specific ActionForward object or not .we can use a method **mapping.findForward("Forwardname")**
syntax:

ActionForward findForward(String forwardName);

when we call findForward() method if the ActionForward object is available it return the reference of that Object. with that name if no ActionForward object is not available it return null value.

```

ActionForward af1 = new ActionForward("failure");
ActionForward af1 = new ActionForward("success");
ActionForward af1 = new ActionForward("one");

```

For the above code it returns ActionForward object for af1 and af2. For af3 it returns a null value.

Developing an Action class which forwards the request to one.jsp

```

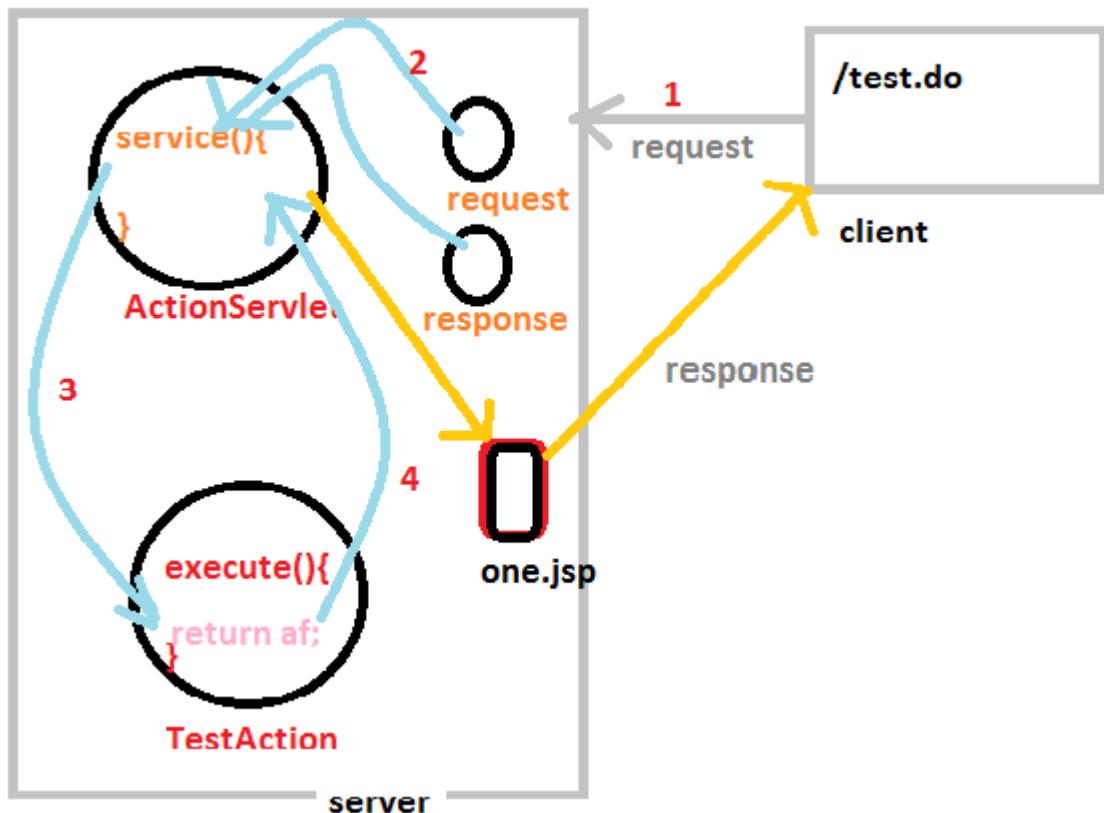
public class TestAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        //storing the arraylist in request scope
        request.setAttribute("name", "sudhakr");
        ActionForward objActionForward = mapping.findForward("success");
        return objActionForward;
    }
}

```

The following steps are carried out when client sends the request to Action class

1. ActionServlet service() creates the Action class object if required.
2. ActionServlet service() finds the path of the Action class in ActionMapping object and call the execute() (ActionServlet service() supplies the ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse objects).

3. server execute execute() method and return ActionForward object to ActionServlet service().
4. Once if the ActionForward object returns to service() method. it checks the path of the resource and It gives input to RequestDispatcher object and it will dispatch the request to appropriate jsp.



Develop a struts based application to retrieve all the records from emp table and display to the client. We should not write any java code in jsps.

we have to follow the following steps:

1. Develop an EmpjavaBean class to store Employee details

```

public class ProductJB implements Serializable {

    Integer pid;
    String pname;
    Double price;

    //Creating the setter methods
    public void setPid(int pid){
        this.pid = pid;
    }

    public void setPname(String pname){
        this.pname = pname;
    }

    public void setPrice(double price){
        this.price = price;
    }

    //Creating the Getter methods
    public int getPid(){
        return pid;
    }
}

```

2. Develop an Action class to interact with database server and retrieve the records and store it in request scope. For that develop one **DAO class** which will return the ArrayList object.

```

public class RetriveProductJB {

    //Retrive data from the table and return the Arraylist object
    public ArrayList getRecords(Integer pid) {
        //code to interact with db
        //store it in arraylist object
        //return the arraylist object
    }
}

```

Now develop the **Action class** as follows.

```

public class TestAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request,
    HttpServletResponse response) {

        RetriveProductJB objRetriveProductJB = new RetriveProductJB();
        ArrayList objArrayList = objRetriveProductJB.getRecords(1);

        //storing the arraylist in request scope
        request.setAttribute("list", objArrayList);

        ActionForward objActionForward = mapping.findForward("success");
        return objActionForward;
    }
}

```

3. The following is the jsp which display output to the client by using core tag libraries.

```

<%@ page isELIgnored="false" %>
<%@ taglib uri="http://" prefix="c" %>

<c:forEach var="emp" items="${emplist}" >
    ${emp.eno}
    ${emp.name}
    ${emp.sal}
</c:forEach>

```

JSTL :

jstl has given sqltaglibrary. This tag library contains set of tags which are used to interact with database server.

It's not recommended to use sql tag library in the jsps . This is because if we use sql tag library we are clubbing business logic and presentation logic.

The following DataSource and update querys are used to write jdbc code in jsp.

```

<%@ page isELIgnored="false" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<sql:setDataSource var="ds" driver="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@localhost:1521:xe" user="hib"
    password="hib"/>
<sql:update dataSource="ds"> insert into emp values(123,
    'sudha',1200)
</sql:update>

```

The following <query> tag is used to display all the records from Emp table.

```
<%@ page isELIgnored="false" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<sql:setDataSource var="ds" driver="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@localhost:1521:xe" user="hib" password="hib"/>
<sql:query var = "rs" dataSource="ds"> select * from emp
</sql:query>
<c:forEach var ="row" items="rs.rows" >
    ${row.eno}
    ${row.name}
    ${row.salary}
</c:forEach>
```

I18N Applications :-

The application which display the output to the client based on their regional language.

We can use property files to develop I18N applications.

I18N applications can be developed for both standalone applications as well as web based applications.

By using properties files we can remove hard coding also.

Every property file must end with an extension called as .properties

In a project we maintain multiple properties files

In a properties file we supply the data in the form of key and value.

eg:

```
key1 = value1
key2 = value2
```

The following is an example demonstrates how to read the contents from properties file.

```
driver_class = weblogic.jndi.WLInitialContextFactory
url = t3://localhost:7001
userName = weblogic
password = harshi0143
dataSourceName = objDataSource
```

DBUtil.properties

Eg:

```
public class DBUtilStandaloneApp {

    // Getting the connection when ever executes this method
    public static Connection getConnection() {
        Connection objConnection = null;
        Properties objProperties = new Properties();
        FileInputStream objInputStream = null;

        try {
            objInputStream = new FileInputStream("DBUtil.properties");
            objProperties.load(objInputStream);

            String driver_class = objProperties.getProperty("driver_class");
            String url = objProperties.getProperty("url");
            String userName = objProperties.getProperty("userName");
            String password = objProperties.getProperty("password");
            String dataSourceName = objProperties.getProperty("dataSourceName");

            // creates an hashtable object
            Hashtable objHashTable = new Hashtable();

            // adding the values to Hashtable
            objHashTable.put(Context.INITIAL_CONTEXT_FACTORY, driver_class);
            objHashTable.put(Context.PROVIDER_URL, url);
            objHashTable.put(Context.SECURITY_PRINCIPAL, userName);
            objHashTable.put(Context.SECURITY_CREDENTIALS, password);

            // Establishing the connection with the directory server
            Context objContext = new InitialContext(objHashTable);

            // getting DataSource object from directory server
            DataSource objDataSource = (DataSource) objContext
                .lookup(dataSourceName);

            // Getting the Connection using DataSource object
            objConnection = objDataSource.getConnection();

            return objConnection;
        }
    }
}
```

The following program for write the contents to Properties file

```

public class WriteToPropsFile {
    public static void main(String[] ar) throws IOException {

        Properties objProperties = new Properties();
        //create file by using FileOutputStream
        FileOutputStream obj FileOutputStream = new FileOutputStream("DBDetails.properties");

        objProperties.setProperty("userName", "sudhakar");
        objProperties.setProperty("password", "sudha");

        objProperties.store(obj FileOutputStream, "dbdetails");
    }
}

```

The following properties file created when we run above program.

```

#dbdetails
#Mon Oct 08 22:34:13 IST 2012
password=sudha
userName=sudhakar

```

DBDetails.properties

We can write the comments in Properties file using # symbol. We can observe that in above example.

In web based applications to read the contents from properties file we are using **getResourceAsStream ("properties filename");** which is available in **Class** class.

To read the properties file by using this method we have to place the properties file in classpath if the program doesn't contain package statement. If the program contains package statement we have to place this properties file inside package folder. we have to place properties files where .classes file is available.

ex: without package in a program we have to keep properties file in classesfolder. Otherwise we have to place that in package folder ex.
blmse/jee/struts/DBUtil.properties.

An organization ISO has assigned the two letter code for every country and every language.

Generally the country letter code will be representing in Capitalse.g.: IN, US

Generally the language letter code will be represent in small letterse.g: te, en

To develop I18N application we required regional and language information of that computer.

To find the current computer regional and language information we can use **java.util.Locale** object.

To find the Locale information of our computer we have to get the Locale object. From the Locale object we can get country code and language code.

ex:

```
public class LocaleDemo {  
    public static void main(String[] ar) {  
  
        //get the Locale object  
        Locale objLocale = Locale.getDefault();  
  
        System.out.println( objLocale.getDisplayCountry() );  
        System.out.println( objLocale.getDisplayLanguage() );  
  
    }  
}
```

To develop I18n applications we need to create multiple properties files. These properties files are called as **Bundle**.

Procedure to use I18N applications in standalone applications:

1. We need to create the properties files. The no. of properties files are based on the languages supported by our project.
In our project we would like to support to 2 languages they are: English and Telugu.

```
hi = u r reading this in Telugu  
  
hello=ya fine yar in Telugu  
  
i18napp_te_IN.properties
```



```
hi = u r reading this in english  
  
hello=ya fine yar in english  
  
i18napp_en_US.properties
```

2. To read the contents from properties file based on client regional language we have to take the help of **java.util.ResourceBundle** class

The following example demonstrates how to read the contents from ResourceBundle.

```

/**It's find the system locale object and sends the output in that language */
public class I18App {
    public static void main(String[] ar) {

        //Getting the System locale object
        Locale objLocale = Locale.getDefault();

        //Getting the Resource Bundle object to read the properties file
        ResourceBundle objResourceBundle = ResourceBundle.getBundle("i18napp",objLocale);

        System.out.println(objResourceBundle.getString("hi"));
        System.out.println(objResourceBundle.getString("hello"));

    }
}

```

When we are retrieving a key from ResourceBundle which is not available it throws an Exception saying:

**Exception in thread "main" java.util.MissingResourceException: Can't find resource for bundle
java.util.PropertyResourceBundle, key hi1**

When we test our application for the properties files which doesn't support other languages application then it throws an Exception saying:

**Exception in thread "main" java.util.MissingResourceException: Can't find bundle
for base name i18napp, locale en_GB**

The project supports two languages and when we run the same project in the computer which doesn't support the other languages we should not get any exception rather than that we should display the default language.

To achieve this from the properties file name we have to remove the country code and language code. ex

hi = u r reading this in english

hello=ya fine yar in english

i18napp.properties

Procedure to use I18N applications in web based applications:

We have developed the following jsp to display the output based on the Client regional language. But it is always displaying output in English language only. This is because we have hardcoded the values in jsp.

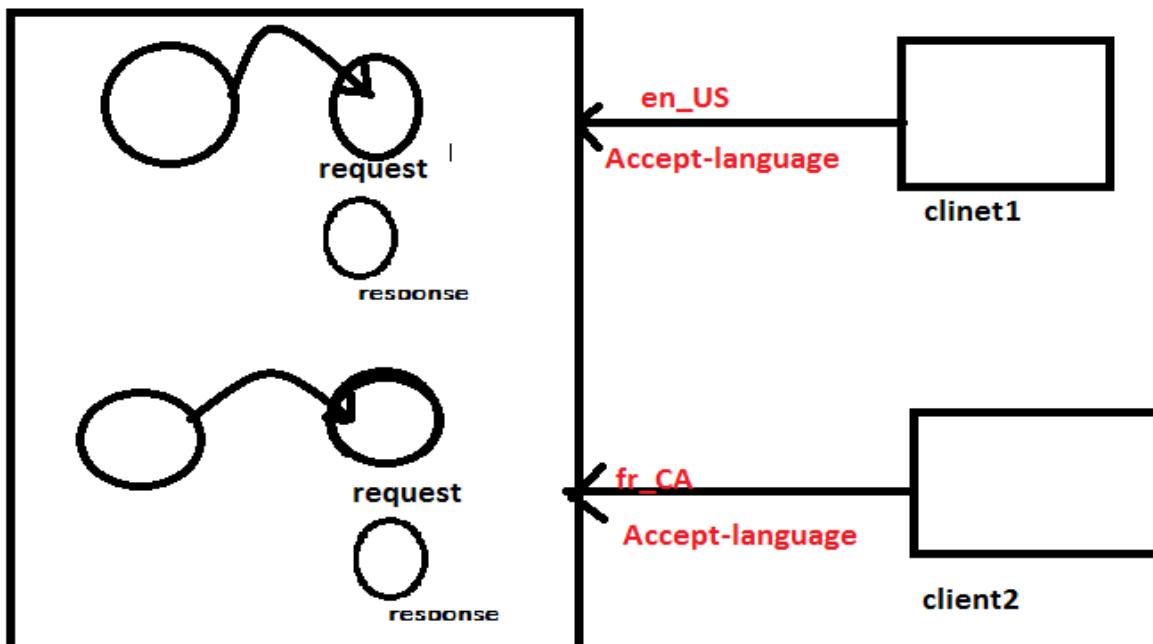
From the following jsp we have to remove the highlighted hardcoded values.

```
<html>
  <head>
    <title>welcome to my project </title>
  </head>

  <body>
    <h1>welcome to java </h1>
    <p>first line </p>
    <p> second line </p>
  </body>
<html>
```

one.jsp

When the client sends the request to a server, Server creates request and response objects. Now server finds the client's regional language from **Accepted-Language** header and creates the Locale object. That Locale object is associated with request object.



The following is the jsp demonstrates that how to do we find the Locale object of client and send it back.

```
<%    Locale objLocale = request.getLocale();
      out.println(objLocale);
%>
```

Procedure to use I18N applications in webbased applications:

1. Create the properties files to support multiple languages as shown below.

```

one=one in english
two= two in english

i18nwebapp_en_US.properties

```

```

one=one in telugu
two=two in telugu

i18nwebapp_te_IN.properties

```

2. Copy all the properties files into classes'folder. This is because all the web based application classpath is set to classes folder.
3. Develop a jsp to find the client regional language and based on that read the contents from properties file and display message to the user.

```

<%@ page import= "java.util.*" %>
<%
    //Getting the Locale object based client regional language
    Locale objLocale = request.getLocale();

    //Reading the data from properties file by using ResourceBundle
    ResourceBundle objResourceBundle = ResourceBundle.getBundle("i18napp",
    objLocale );

%>
<h1><%=objResourceBundle.getString("one") %>
<h2><%= objResourceBundle.getString("two") %>

```

one.jsp

The disadvantage of above jsp is we have provided lot of java code. It's not recommended to write java code in Jsp's. To remove the java code from above jsp we have to use jstl tag library as shown below.

```

<%@ page import= "java.util.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<html>
<head>
<title>JSTL fmt:bundle Tag</title>
</head>
<body>
<fmt:bundle basename="i18nwebapp">
    <fmt:message key="one"/><br/>
    <fmt:message key="two"/><br/>
</fmt:bundle>
</body>
</html>

```

```

<fmt:setBundle basename="i18nwebapp" var="lang"/>

<fmt:message key="one" bundle="${lang}"/><br/>
<fmt:message key="two" bundle="${lang}"/><br/>

```

Procedure to use I18N applications inStruts based applications:

1. Create the properties files to support multiple languages as shown below.

one=one in english
two= two in english

i18nwebapp_en_US.properties

one=one in telugu
two=two in telugu

i18nwebapp_te_IN.properties

2. Copy all the properties files into classes folder.This is because all the web based application classpath is set to classes folder.
3. Configure the properties files in struts-config.xml file by using **<message-resources>** tag.
ex:
<struts-config>
 <message-resources parameter="messages" />
</struts-config>
4. Develop a jsp to display the output to the user based on the struts tag library. Use **message tag of bean tag library**.as shown below

```

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<bean:message key="one" /><br/>
<bean:message key="two" /><br/>

```

When we use the **message tag** for a key which is not available it will throw anexception saying **Missing message key for bundle**.

Generally when multiple people are working on same project and they want to share properties files andwhenever we are using a key which is not available my application

should not throw any exception. To achieve this we have to configure **null=false** in the struts configuration file as shown below

```
<struts-config>
    <message-resources parameter="messages" null = "false"/>
</struts-config>
```

By default struts uses **null="true"** the meaning of null= true is if the key is not available it returns a null value to struts. When it returns null value to struts it throws an exception. When we specify null=false even if the key is not available it will not throw any exception it throws the following error.

eg: **???en_US.msg.ls???**

It is nothing but client regional language and which key is not available.

Using multiple bundles in struts:

We are trying to use multiple bundles in struts based projects. As shown below.

1. Develop the properties files for multiple bundles as shown below

```
userName = sudhakar in english
password = abcdefg in english
```

resone_en_US.properties

r
e
s
o
n
e

```
userName = sudhakar in french
password = abcdefg in french
```

resone_fr_CA.properties

r
e
s
o
n
e

```
myname = madhav in english
darling = laxmi in engilsh
```

restwo_en_US.properties

r
e
s
t
w
o

```
myname = madhav in french
darling = laxmi in french
```

restwo_fr_CA.properties

3. Copy all the properties files into classes folder.This is because all the web based application classpath is set to classes folder.
4. Configure the properties files in struts-config.xml file by using **<message-resources>** tag.

ex:

```

<struts-config>
    <message-resources parameter="resone" />
    <message-resources parameter="restwo" />
</struts-config>

```

5. Develop a jsp to display the output to the user based on the struts tag library. Use **message** tag of **bean** tag library. as shown below

```

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>

Username :<bean:message key="userName" /><br>
password :<bean:message key="password" /><br>

fathername :<bean:message key="myname" /><br>
mothername<bean:message key="darling" /><br>

```

i18napp.jsp

When we run above program it will throw an exception Missing key for resource **userName** and **password**. we have observed that for the above configuration in struts configuration file it is able to read the contents from only one resource bundle that is which is configured as last file in struts configuration file. in our example it is able to read the messages from **onlyrestwo** bundle. When we are reading keys in **resone** bundle server throwing an exception.

To resolve the above the problem we have made changes in struts configuration file by adding an attribute **keyinmessage-resources** tag.

```

<struts-config>
    <message-resources parameter="resone" key="rone"/>
    <message-resources parameter="restwo" key="rtwo"/>
</struts-config>

```

While we are using the **message** tag in the jsp we need to specify in which bundle It have to search by using an **attribute** called as **bundle**.

```

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>

<bean:message key="one" bundle="msgs"/><br>
<bean:message key="two" bundle="msgs" /><br>

Username :<bean:message key="userName" key="rone" /><br>
password :<bean:message key="password" key="rone" /><br>
fathername :<bean:message key="myname" key="rtwo" /><br>
mothername<bean:message key="darling" key="rtwo" /><br>

```

i18napp.jsp

Explanation:

When we configure ResourceBundles without keys they try to store two bundles into application scope (ServletContext object) by using same key as **org.apache.sturts.action.Message** whenever server has encountered **<message-resources>** tag Server is creating a **org.apache.struts.util.PropertyMessageResources** object.

```

<struts-config>
    <message-resources parameter="resone" />
    <message-resources parameter="restwo" />
</struts-config>

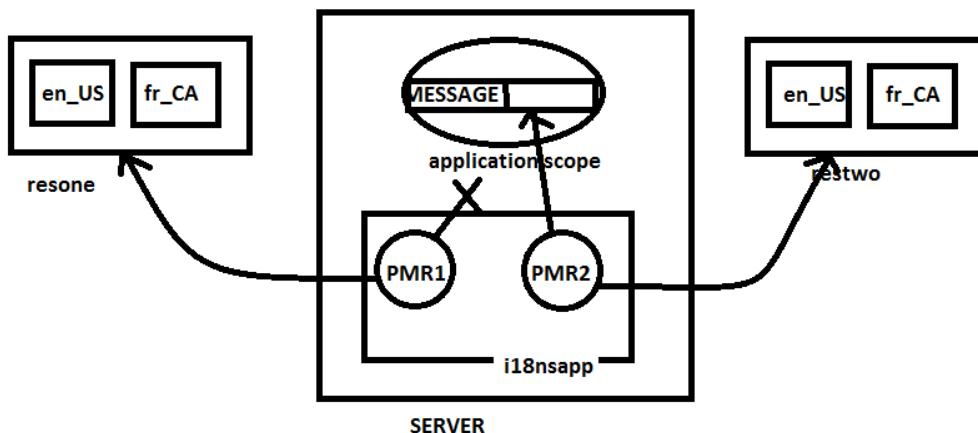
```

When the above configuration is executed it will try to create two **PropertyMessageResources** objects. And trying to store these objects In application scope with default key (**org.apache.sturts.action.Message**).

eg:

```
application.setAttribute(Globals.MESSAGE_KEY, objPropertyMessageResources1);
application.setAttribute(Globals.MESSAGE_KEY, objPropertyMessageResources2);
```

in this example **objPropertyMessageResources2** is overrides the first value. that's why we are able to read the values from only one ResourceBundle.



When we modify the properties files we need restart the server. Because properties files read by server only once. ie at the time of deployment.

org.apache.sturts.action.Message value is available as key for **Globals.MESSAGE_KEY**. The logic tag library contains set of tags which are used to check the logical conditions. for example Present, notpresent, greaterthan and lessthanetc.

```
<logic:present name="Globals.MESSAGE_KEY" scope="application">
```

The key is available

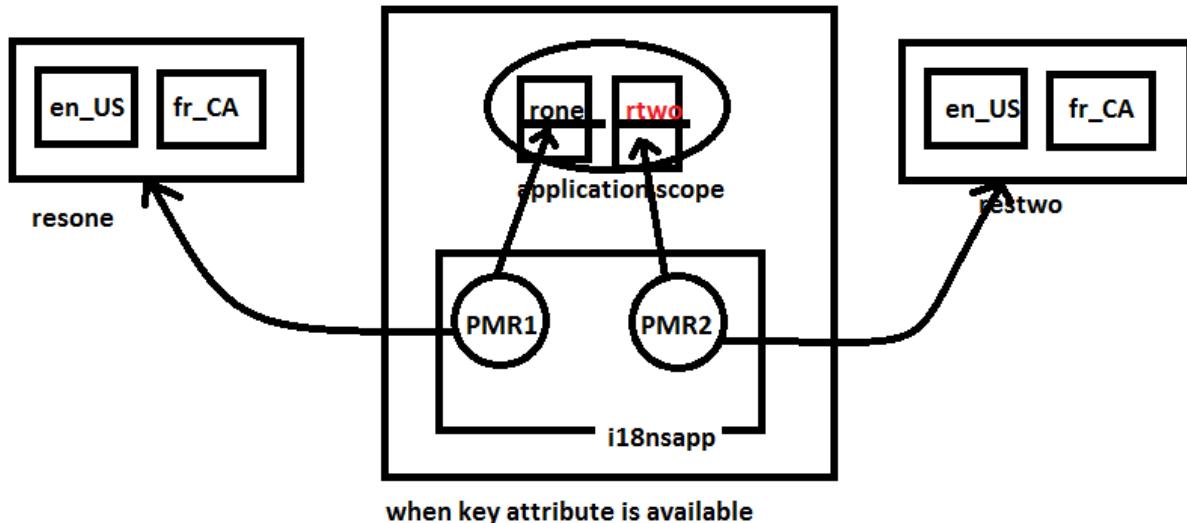
```
</logic:present>
```

When we specify key attribute in message-resources then with that key

PropertyMessageResources object will be stored in application scope as shown below

```
application.setAttribute("rone", objPropertyMessageResources1);
```

```
application.setAttribute("rtwo", objPropertyMessageResources2);
```



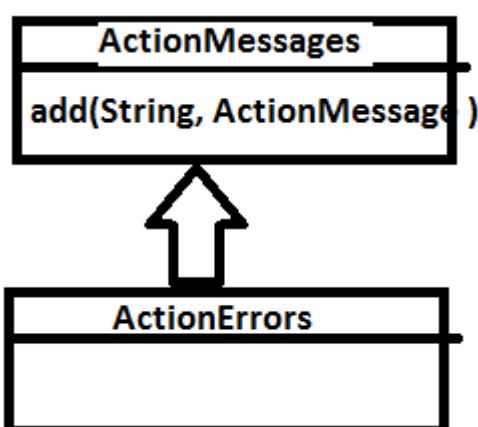
How to deal with errors in struts?

In struts 1.2 they have given 4 different classes to deal with errors/Messages . They are

1. ActionErrors
2. ActionError
3. ActionMessages
4. ActionMessage

But in struts 1.3 they removed the ActionError class .so in struts 1.3 we have only three classes to deal with errors.

If we want to deal with ActionErrors we have to create ActionErrors object and add Error message by using ActionMessage object.



Requirement : Develop an action class which generates 5 errors and display it in the jsp.

Step1: create the properties files and add all the errors to properties files.then place those properties files in classes folder then configure that in the struts configuration file.

ex:

```
userName = sudhakar  
password = sudha0143  
address = karur  
phone = 9491919112
```

MessageResources.properties

step2: Develop an action class which generates errors and add it to the requestscope.

```
public class TestErrorAction extends Action {  
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request,  
                                HttpServletResponse response) {  
        //creating ActionErrors object to display the errors  
        ActionErrors objActionErrors = new ActionErrors();  
        //creating ActionMessage objects and supply the error key configured in properties file  
        ActionMessage objActionMessage1 = new ActionMessage("userName");  
        ActionMessage objActionMessage2 = new ActionMessage("password");  
        ActionMessage objActionMessage3 = new ActionMessage("address");  
        ActionMessage objActionMessage4 = new ActionMessage("phone");  
        //adding all message objects to ActionErrors object  
        objActionErrors.add("e1",objActionMessage1);  
        objActionErrors.add("e1",objActionMessage2);  
        objActionErrors.add("e2",objActionMessage3);  
        objActionErrors.add("e3",objActionMessage4);  
        saveErrors(request, objActionErrors);  
    }  
}
```

TestErrorsAction

Step3: Configure Action class into struts configuration file. As shown below

```
<action-mappings>  
    <action path="/error" type="TestErrorAction" >  
        <forward name="done" path="/pages/error.jsp" />  
    </action>  
</action-mappings>
```

struts-config.xml

Step4: Display errors in one.jsp

To display errors in struts we use <errors>tag in html tag library.

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>  
< html:errors />
```

one.jsp

Explanation:

When client sends the request to the server to execute TestErrorAction the following steps are carried.

step1: Server has handover request object response object to the ActionServlet service() method. ActionServlet service() creates the Action class object if required.

step2:ActionServlet service() finds the path of the Action class in ActionMapping object and call the execute() (ActionServlet service() supplies the ActionMapping, ActionForm, HttpServletRequest, HttpServletResponse objects).

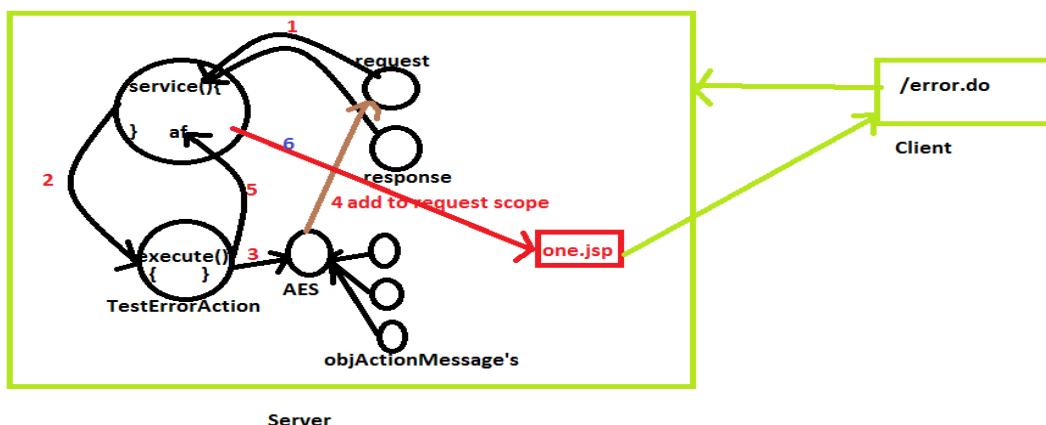
step3:execute() method create ActionErrors object and add the ActionMessage object's to that. ActionMessage is taking the **error key** as parameter.

Step4:When the **saveErrors(aes, request)** method is called ActionErrors object will be stored in request scope by using a key **org.apache.struts.action.ERROR**

Step5:server execute execute() method and return ActionForward object to ActionServlet service(). Once if the ActionForward object returns to service() method. it checks the path of the resource in ActionForward object and It gives input that to RequestDispatcher object and it will dispatch the request to **error.jsp**.

Step6:**One.jsp** displays the errors.

when the **<html:errors />**tag is evaluated it will get error key from request scope and get all the error messges from the properties files and display to the client.



To display error messages we are using a tag **<html:errors />** This tag uses html tags which are configured in properties file. The following are the tags which are available in properties file.

```
# -- standard errors --
errors.header=<UL>
errors.prefix=<LI>
errors.suffix=</LI>
errors.footer=</UL>
```

It's recommended to use **<html: messages>** tag to display errors

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<ul>
<html:messages id="msg" >
    <li><bean:write name="msg" /></li>
</html:messages>
</ul>
```

error.jsp

To display the specific errors based on a group we **use property attribute** as shown below. Here **e1** is group name with which we stored ActionMessage object into ActionErrors object. eg: `aes.add("e1", am1);`

```
<html:messages id="msg" property="e1">
    <li><bean:write name="msg" /></li>
</html:messages>
```

We achieve same thing using `<html:errors />` tag also. eg:

```
<html:errors property="e2" />
```

In Struts we can use any URL pattern in web.xml file. foreg: we can use ***.htm** for Action servlet as shown below.

```
<web-app>
    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.htm</url-pattern>
    </servlet-mapping>
</web-app>
```

web.xml

Generally struts people follow any of the following URL patterns .

1. *.do
2. /do/*

eg: as shown below configuration in web.xml

```

<web-app>
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
    <url-pattern>/do/*</url-pattern>
  </servlet-mapping>
</web-app>

```

web.xml

When client sends the request to server which ends with .do or which start with /do/ then that URL matches to ActionServlet so ActionServlet service() method will be executed.

- eg:
1. login.do
 - 2 /do/login

In both examples it will invoke the login Action class whose **path is login** configured in struts-config.xml in ActionMappings tag.

If we want to work with Messages in struts we have to **create ActionMessages object** and add the messages to ActionMessage object then add the ActionMessage object to ActionMessages object .eg:

```

ActionMessage am= new ActionMessage("title");
ActionMessages ams = new ActionMessages();
ams.add("msg1", am);

```

property name it is used to display messages based on this

message key configured in properties file

Requirement : Develop an action class which generates 5 messages and display it in the jsp.

Step1:create the properties files and add all the messages to properties files. then place those properties files in classes folder then configure that in the struts configuration file.

```

title= struts messge app
header= welcome to struts app
body= we are displaying messages
footer = sudhakar company.

```

MessageResources.properties

step2:Develop an action class which generates messages and add it to the requestscope.

```
public class TestMessageAction extends Action {  
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request,  
                                HttpServletResponse response){  
        ActionMessages objActionMessages = new ActionMessages();  
        ActionMessage objActionMessage1 = new ActionMessage("title");  
        objActionMessages.add("title", objActionMessage1);  
        objActionMessages.add("header", new ActionMessage("header"));  
        ActionMessage objActionMessage3 = ;  
        objActionMessages.add("header", new ActionMessage("body"));  
        objActionMessages.add("footer", new ActionMessage("footer"));  
        saveMessages(request, objActionMessages);  
        return mapping.findForward("messages");  
    }  
}
```

TestMessagesAction

When `saveMessages()` method is executed all messages are stored into request scope by using a key `org.apache.struts.action.MESSAGE`

Step3: Configure Action class into struts configuration file. As shown below

```
<action-mappings>  
    <action path="/messages" type="TestMessageAction" >  
        <forward name="messages" path="/pages/messages.jsp" />  
    </action>  
</action-mappings>
```

struts-config.xml

Step4: Display messages in messages.jsp

To display messages in struts we use `<messages>` tag in html tag library as shown below.

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>  
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>  
<html:messages id="msg" message="true" >  
    <li><bean:write name="msg" /></li>  
</html:messages>
```

messages.jsp

By default `messages` tag get the data from `ERROR KEY`. To get the data from `org.apache.struts.action.MESSAGEKEY` we must specify `message=true` attribute. For `<html: messages >` tag we can use `property` attribute to display based on the location or group.

eg: `<html: messages id="msg" property="header" message="true">`

In program it self we can create the ActionForward object and give an input to the jsp file to which jsp we want to forward the request.

```
ActionForward af = new ActionForward("/one.jsp");
```

It's not recommended this approach. This is because we are hard coding the path of the resource.

Form based application in Struts :

Majorly in struts we have 3 ways to develop the form based applications: They are :

1. ActionForm
2. DynaActionForm
3. DynaValidatorForm

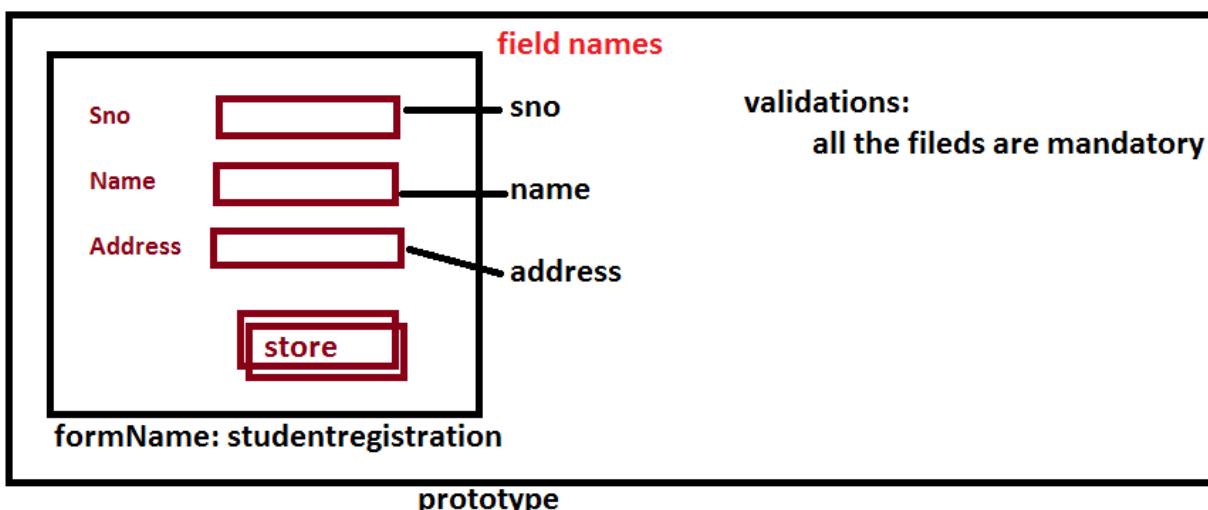
Form based application using ActionForm :

we have to follow the following steps to develop form based apps

Step1 :Get the Prototype. From the prototype identify the following three elements

1. Identify the field names (Property name) for every field available in that form.
2. Identify the form name (the name shouldn't contain any spaces)
3. Identify the all the validations required on that form.

eg:



Step2: Develop the FormBean class

What is FormBean class?

A class which is a subclass of org.apache.struts.action.ActionForm directly or indirectly

```
org.apache.struts.action.ActionForm  
void reset(ActionMapping ,HttpServletRequest)  
// Can be used to reset bean properties to their default  
state, as needed.  
ActionErrors validate(ActionMapping,HttpServletRequest)  
//Can be used to validate the properties
```



StudentFB

```
//provide the setters and getter for identified by  
properties in step1.  
//override the reset and validate() methods
```

Developing StudentFB class with 3 properties and override the reset() and validate() methods as shown below.

```

public class StudentFB extends ActionForm {
    String sno;
    String name;
    String address;

    public StudentFB(){
        System.out.println("StudentFB Object is created");
    }

    public void setSno(String sno){
        System.out.println("StudentFB setsno() called");
        this.sno = sno;
    }

    public String getSno(){
        System.out.println("StudentFB getsno() called");
        return sno;
    }
    // provide setters and getters for all remaining fields

    //to reset the fields
    public void reset(ActionMapping mapping, HttpServletRequest request){
        setSno("Enter sno");
        setName("Enter Name");
        setAddress("Enter address");
        System.out.println("we are in reset()");
    }
    public ActionErrors validate(ActionMapping mapping, HttpServletRequest
                                request){
        ActionErrors actionErrors = new ActionErrors();
        System.out.println("we are in validate()");

        //check is the filesd are empty if empty add the errors
        if (sno == null || sno.equals("")){
            actionErrors.add("sno", new ActionMessage("sno.required"));
        }
        if (name == null || name.equals("")){
            actionErrors.add("name", new ActionMessage("name.required"));
        }
        if (address == null || address.equals("")){
            actionErrors.add("address", new ActionMessage
                ("address.required"));
        }
        return actionErrors;
    }
}

```

Step3: configure the FormBean class in struts configuration file

1. copy all the formbean classes into classes folder.
2. provide all the error messages in properties files and configure them in struts configuration file.

```
sno.required = Student no is required  
name.required = Name is required  
address.required = Address is required
```

student_en_US.properties

3. configure the Formbean class in struts configuration file in <form-beans>tag

```
<struts-config>  
    <form-beans>  
        <form-bean name="studentRegForm" type="StudentFB" />  
    </form-beans>  
    ...  
    <message-resources parameter="student" null="false"/>  
</struts-config>
```

struts-config.xml

Step4:Develop the input form or form view

The following form responsible to display the form as like prototype which uses html tag library.

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>  
  
<html:html>  
    <head>  
        <title> my struts first form app</title>  
    </head>  
  
    <body>  
        <center>Student registrationform</center>  
        <html:errors />  
        <html:base />  
        <html:form action="/studentreg.do" ><br>  
            Sno: <html:text property="sno" /><br>  
            Name: <html:text property="name" /><br>  
            Address: <html:text property="address" /><br>  
  
            <html:submit value="store"/>  
            <html:reset/>  
        </html:form>  
    </body>  
</html:html>
```

studentregform

Step5:Develop an action class which is responsible to capture the data and store it database server.

The struts internal code is responsible to create FormBean object and capture the data and store it in FormBean object. This formbeanobject supplied to parameter to execute() method .

In the execute() method from the actonform object we get the data and store in database server.

```
public class StudentAction extends Action {  
    public StudentAction(){  
        System.out.println("StudentFB Object is created");  
    }  
  
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest  
                                request, HttpServletResponse response) {  
        //Get the details from FormBean object  
        StudentFB studentFB = (StudentFB) form;  
  
        //code to store the data in database server  
        System.out.println("student sno: " +studentFB.getSno() );  
        System.out.println("student name: " +studentFB.getName() );  
        System.out.println("student address: " +studentFB.getAddress());  
  
        return mapping.findForward("success");  
    }  
}
```

StudentAction.java

Step6:Configure the action class in struts configuration file

```
<action-mappings>  
    <action path="/studentreg" type="StudentAction" name="studentRegForm"  
           scope="request" validate="true" input="/studentregform.jsp" >  
  
        <forward name="success" path="/success.jsp" />  
    </action>  
</action-mappings>
```

Struts-config.xml

Step7: develop a success form (success view).

This is responsible to display success message to the user.

```
hi successfully stored in db....<br>  
<a href="studentregform.jsp">register again here</a>
```

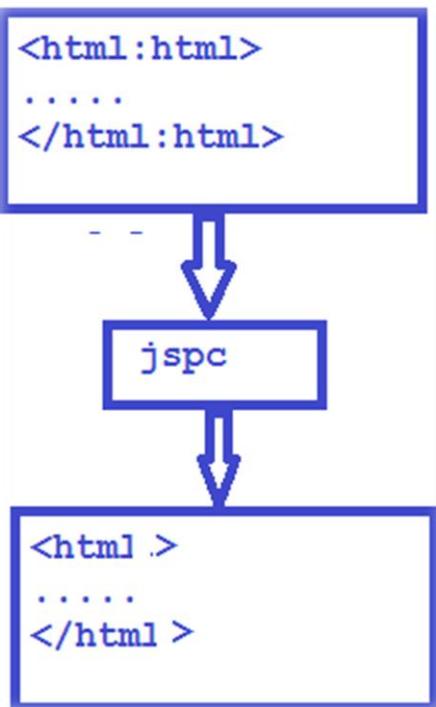
Success.jsp

Deploy the project and test it. Once we have deploy the above project we have to start testing the project from the jsp(studentregform.jsp).

When we starting executing the studentregform.jspwe have observed that the internal code is creating FormBean object and doing a lot of other work.

Explanation:

When we send the request to the server for studentregform.jsp server starts executing that. As part of that when server encounters <html:html> it will be evaluated by jsp compiler has converted into regular <html> tag.



We can provide an attribute called as lang to the html tag. This attribute will find the language setting of the client and send it back. The advantage of this attribute is to fix some language specific problem.

<html:html lang="true" />

When the jsp compiler encountered head, title, body tags it is treated as template text and sent to the client.

<html:form >

To create the form in jsp we use <html:form> tag. Whenever we are using <html:form> tag without action attribute it throwing an exception.

```
<html:form>\n</html:form>
```

The above tag will throw an NullPointerException. this is because we doesn't supply the action attribute to the form tag.

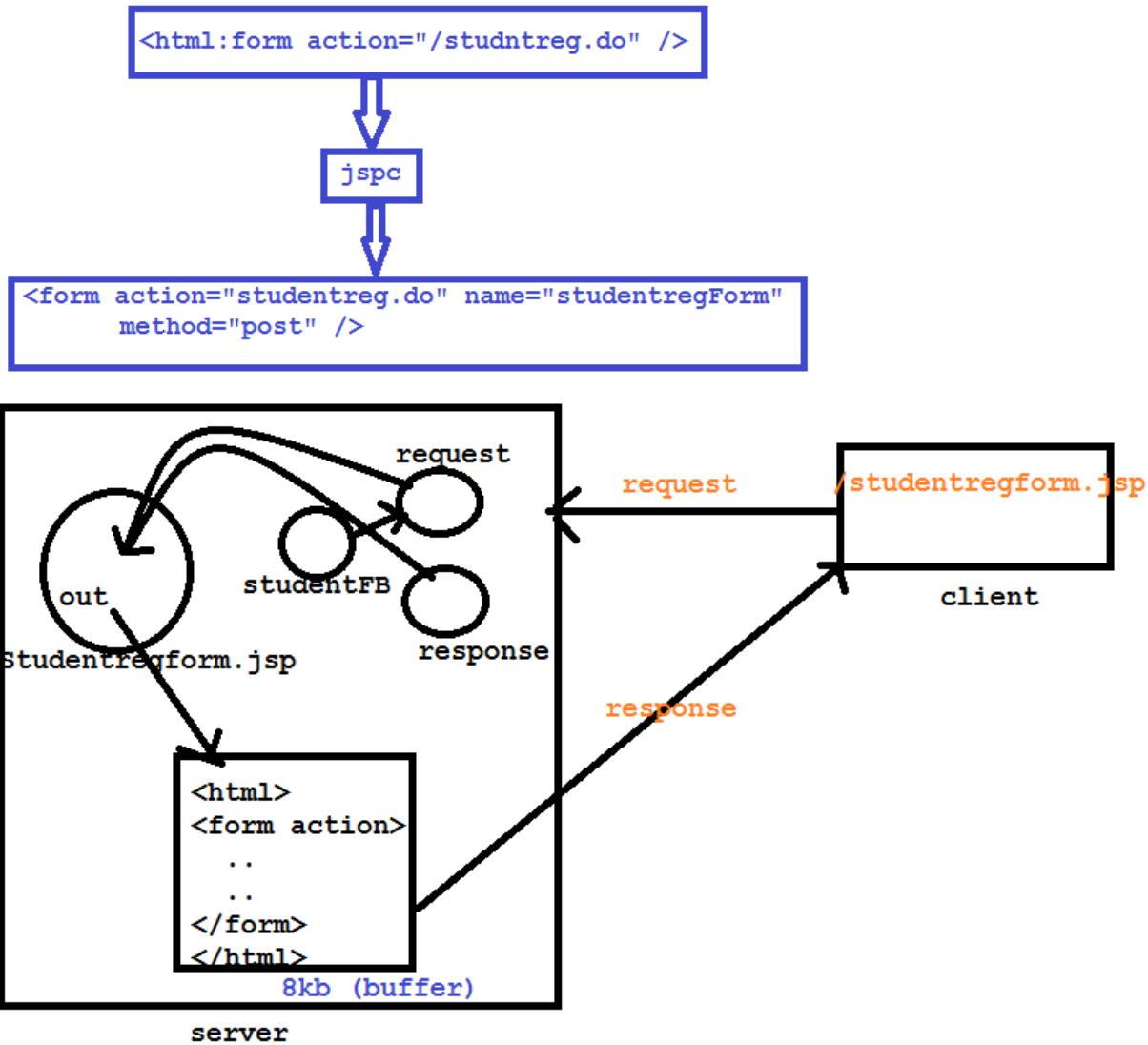
When <html: form> tag is evaluated (evaluating the java code) it takes the value of the action attribute. From the value it will remove the .do if it is available. Now we got the path of the resource. Now struts internal code checks is there any ActionMapping object is available whose path matches to it. If ActionMapping object is not available it throw an Exception saying : Can't retrieve mapping for action

If ActionMapping object is available it gets the name from ActionMapping object. For that name Struts internal code checks is there any FormBean object is available whose name matches to it. If not available it throw an Exception saying:

Matching FormBean object is not available.

If matching formBean object is available then it gets the name of the FormBean class. Now struts internal code checks what is specified scope attribute in action tag. In that scope it will check whether FormBean object is available or not. If FormBean object is not available then it's creates the FormBean object using Class.forName(). After creating the FormBean object it will call the reset() method. This FormBean object is added to specified scope.

Now html:form tag will sent regular html form tag to the client.



From the above diagram we have understand that FormBean object is added to request scope. When the server sends the output to the client immediately server removes request and response objects. Now the server remove the FormBean object also when scope is request.

If we don't specify the scope attribute in configuration file by default the scope it uses as Session. `scope="session"` ---→ default value for scope attribute.

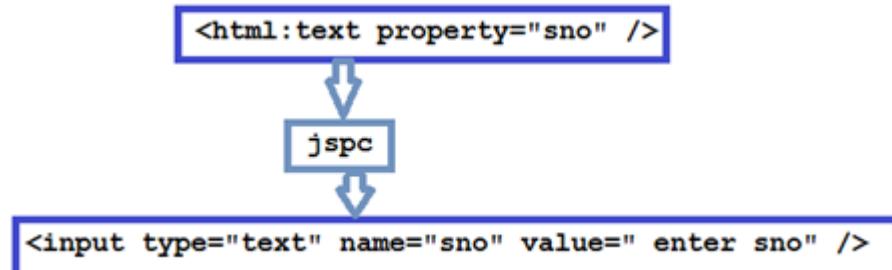
`<html:text />`

Whenever `<html:text/>` is evaluated the following steps will be carried out:-

1. It checks whether the mandatory attribute property is provided or not. If not available send error messages.

2. If the property attribute is available it gets the value of property attribute and check whether this property is available in FormBean object or not? If not available it sends an error message.
3. if matching property is available then struts internal code will call the getXxx() methods.

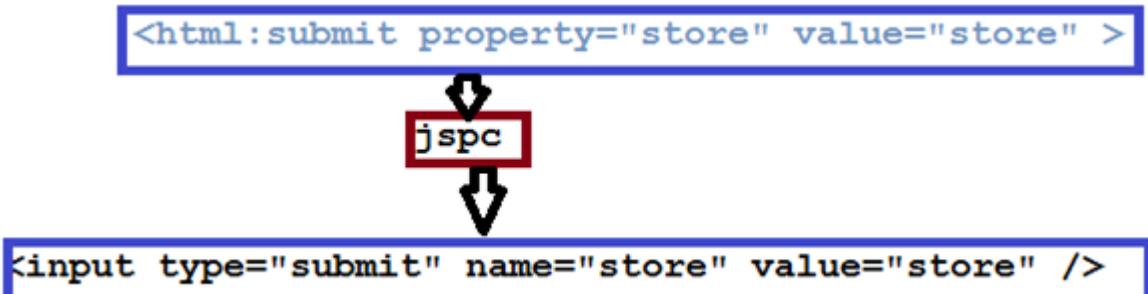
The struts internal code uses return value of getter () methods and generates regular html tags as shown below.



<html:submit />

When the above tag is evaluated it will generate regular submit button.

For submit button we no need to provide the property attribute. If we supply property as attribute it takes property as name. This attribute is used to uniquely recognize the button.

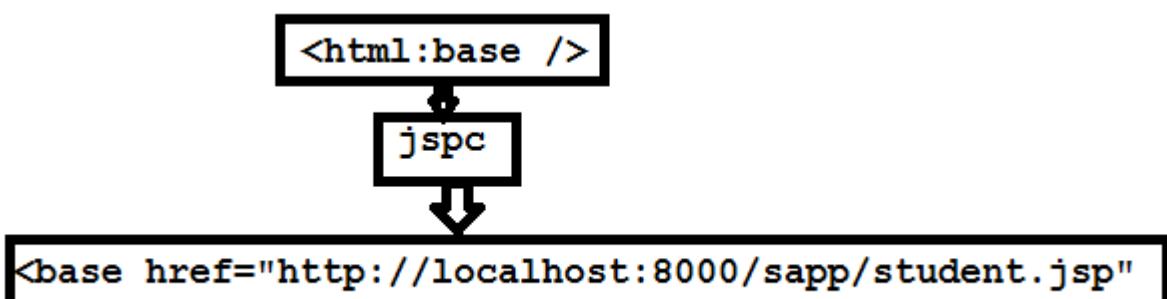


<html:errors />

When above tag is evaluated it will check any errors are available in request scope or not. If not available it will not display any errors. If it is available it will display errors to the user.

<html:base />

When this tag is evaluated it will find the corresponding jsp which is getting executed and it gives that URL as value to the base tag. The advantage of this is the maintenance of project becomes easy.



For the <html: form />tag we can supply the attributes like disabled, focus, and onsubmit attributes. By using onsubmit attribute we can perform client side validations.

disabled="true" :To disable the form .

ex:<html:form action="sa.do" disabled="true" /> then form is not available to do any operation

focus="propertyName" :To focus on the element or put cursor on specified field

<html:form action="sa.do" focus="sno"/>

onsubmit="return validate()" :

```
<script>
    function validate(){
        var name= document.forms[0].name.value;
        var address = document.forms[0].address.value;
        if(sno =="" || sno.match("Enter sno")){
            alert("enter sno");
            return false;

        }
        if(name == ""){
            alert("enter name");
            return false;
        }
        if(address == ""){
            alert("enter address");
            return false;
        }
    }
</script>
<html:form action="studentreg" focus="sno" onsubmit="return validate()"><br>
    Sno    <html:text property="sno"/><br>
    Name <html:text property="name"/><br>
    Address <html:textarea property="address"/><br>

<html:submit value="store" property="store" />
```

If we want to make sure that the form is submitted only once, The user clicks on submit button we need to disabled submit button so that form is submitted only once.

```
<script>
    function validate(){
        document.forms[0].store.disable="true";
        return false;
    }
</script>
```

submit button name. we provide the help of property attribute

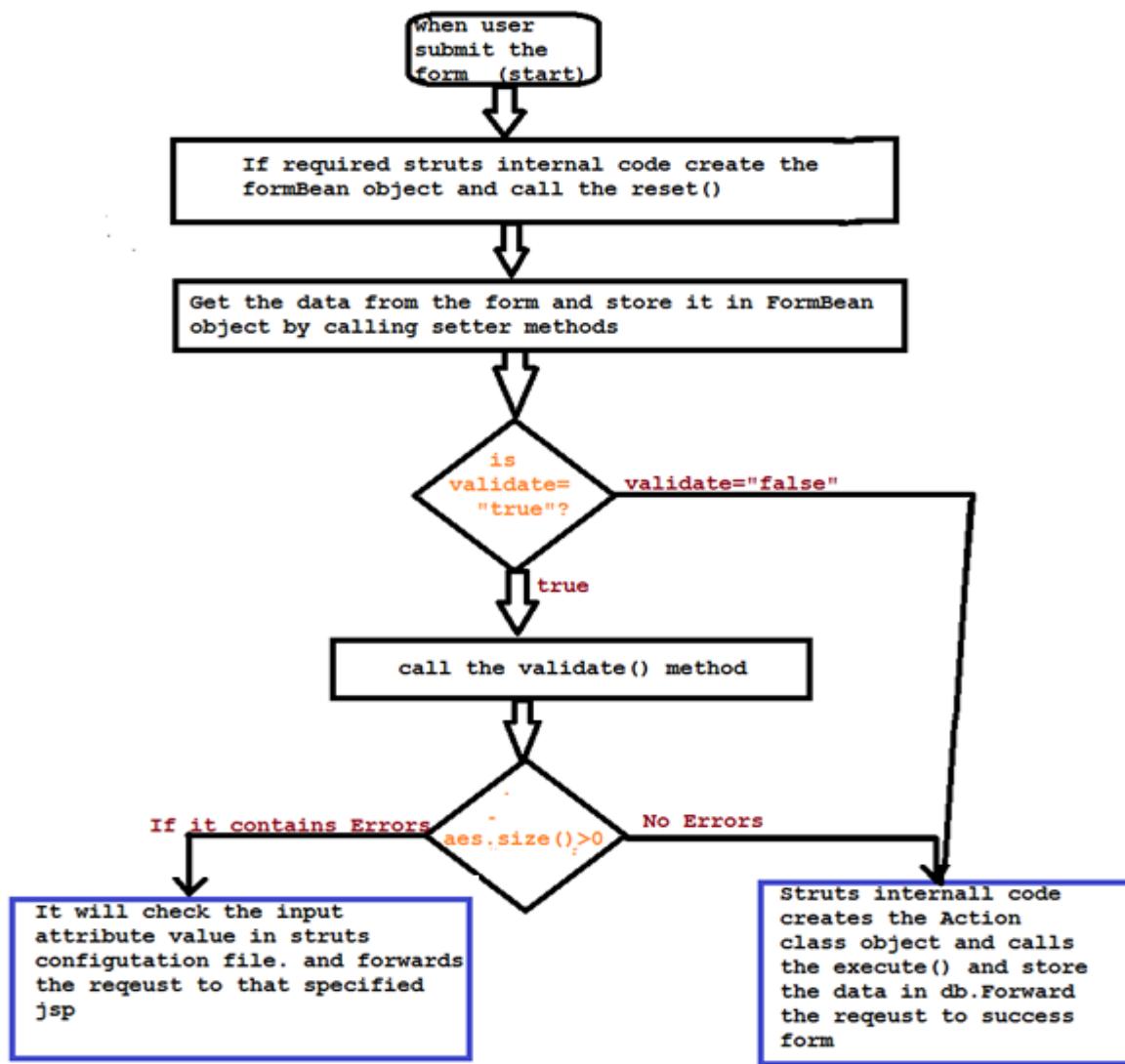
The following steps will be carried out when we submit the form: ---

The request will go to the server. Request and response objects will be created then ActionServletservice () method will execute as part of this it will check is there any name attribute. If it finds the name attribute then it gets the value of that with that name it will check is there any FormBean object is configured if it finds then it will get the FormBean class name.

- 1) Now server will check is there any FormBean object is available in specified scope. If required the struts ActionServlet service() (struts internal code) Create FormBean object and call the reset().
 - if scope="request" every time when we submit the form every time it will create FormBean object.
 - If scope="session" FormBean object is created only once.
- 2) Struts internal code (ActionServlet service ()) will capture the data and store the data into FormBean object by calling setXxx() methods.

Note: Internally struts uses request.getParameterNames () and reflection API to store the values to FormBean object.

- 3) Struts internal code (ActionServlet service()) checks whether validate attribute is available or not in struts configuration file. If we are not writing any validate attribute in configuration file then the default value for validate is true from struts 1.3
Before that validate ="false" up to struts 1.2 .
 - If value is true it will call the validate () method of FormBean class.
 - if validate="false" it will not call validate () method. Struts internal code (ActionServlet service() method) will create Action class object and call the execute() method. execute() method is responsible to store the data into database server and forward the request to appropriate success form.
- 4) if the validate() method returns errors to ActionServlet service() method then it will check the input attribute value specified in <action > tag and get's the value and forward the request to student.jsp.
<action name="a" input="/student.jsp" />
 - If there are no errors return by validate() method to ActionServlet service() method then struts internal code(ActionServlet service() method)will create Action class object and call the execute() method.
 - execute() method is responsible to store the data into database server and forward the request to appropriate success form.



Validations:

We can perform two types of validations on the form. They are:

1. Server side validations.
2. Client side validations.

Server side validations:-

- The server side validations will be carried out in the server. The disadvantage of this is it consumes more resources of server. The advantage is we write the validation code in java.
- If we are performing server side validations if multiple clients send the request at the same we can find more load on the server. To resolve this type of operation we can use client side validations.

Client side validations :-

The client side validations will be carried out in the browser itself. For client side validations we will write in JavaScript.

```

<script>
    function validate() {
        document.forms[0].store.disable="true";
        return false;
    }
</script>

```

submit button name. we provide the help of property attribute

We want to develop a form based application where we want to get city and state data from the user. But the information about city and state available in a table called as city-state table. In this example if we want to perform client side validations we have to write huge amount of JavaScript code to carryout validations. In this type of scenarios not recommended to use client side validations.

When do we use Server side validations?

If we want to check the data entered by the user with the data available in database server we have to use server side validations.

When do we use Client side validations?

If we want to check the data entered by the user is not dependant on the data which is available in database server we have to use client side validations.

In a project if we want we can carry out both server side validations and client side validations.

Form based application using DynaActionForm :

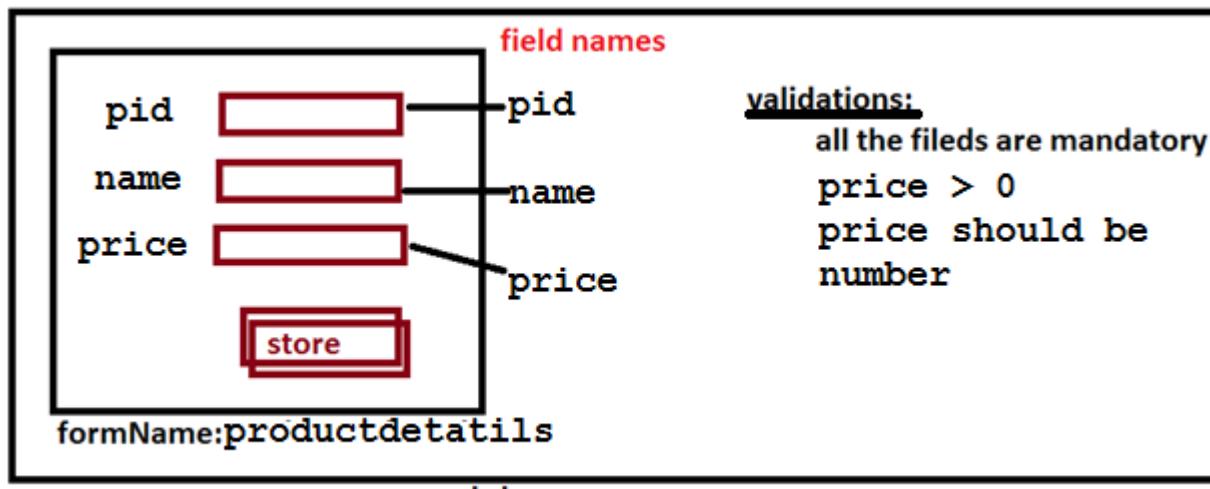
We have to follow the following steps to develop form based apps

When we are working with DynaActionForm we no need to write setter() and getter() methods in FormBean class. Internally it is implementing to DynaBean as part of this interface it is providing the setter and getter() methods. DynaBean is available in commons-beanutils.jar file.

Step1 :Get the Prototype. From the prototype identify the following three elements

4. Identify the field names (Property name) for every field available in that form.
5. Identify the form name (the name shouldn't contain any spaces)
6. Identify all the validations required on that form.

eg:



Step2: Develop the FormBean class based on DynaActionForm

What is FormBean class?

A class which is a subclass of org.apache.struts.action.ActionForm directly or indirectly

```
org.apache.struts.action.ActionForm

void reset(ActionMapping ,HttpServletRequest)
// Can be used to reset bean properties to their default
state, as needed.

ActionErrors validate(ActionMapping,HttpServletRequest)
//Can be used to validate the properties
```



```
org.apache.struts.action.DynaActionForm

set(propertyName, objectValue)
get(propertyName)
```



```
ProductFB

//override the reset and validate() methods
```

Developing ProductFB class :

informbean class override the reset() and validate() methods as shown below.

To store the data into a property we use setter() method. To get the data from property we use getter().

Syntax: void set(String propertyName, Object value)

Object get(String propertyName)

```

public class ProductFB extends DynaActionForm {
    public void reset(ActionMapping mapping, HttpServletRequest request){
        System.out.println("we are in reset()");
        set("pid", "enter pid");
        set("name", "enter name");
        set("price", "0");
    }
    public ActionErrors validate(ActionMapping mapping, HttpServletRequest request){
        System.out.println("we are in validate()");
        String pid = (String) get("pid");
        String name = (String) get("name");
        String price = (String) get("price");
        //creating ActionErrors object
        ActionErrors objActionErrors = new ActionErrors();

        if(pid == null || pid.equals ""){
            objActionErrors.add("pid", new ActionMessage("pid.required"));
        }
        if(name == null || name.equals ""){
            objActionErrors.add("name", new ActionMessage("name.required"));
        }
        if(price == null || price.equals ""){
            objActionErrors.add("price", new ActionMessage("price.required"));
        } else {
            Double cprice = Double.parseDouble(price);
            if(cprice < 0){
                objActionErrors.add("price", new ActionMessage("price.minval"));
            }
        }
        return objActionErrors;
    }
}

```

Step3: configure the FormBean class in struts configuration file

4. copy all the formbean classes into classes folder.
5. provide all the error messages in properties files and configure them in struts configuration file.

```

pid.required="Product id is required"
name.required=Product name is required
price.required=Product price is required
price.minvalue= Price should not less than 0

```

MessageResources.properties

6. configure the Formbean class in sturts configuration file in <form-beans>tag

```

<form-beans>
    <form-bean name="productDetailsForm" type="ProductFB">
        <form-property name="pid" type="java.lang.String" />
        <form-property name="name" type="java.lang.String" />
        <form-property name="price" type="java.lang.String" />
    </form-bean>
</form-beans>
...
<message-resources parameter="MessageResources" null="false" />

```

struts-config.xml

Step4:Develop the input form or form view

The following form responsible to display the form as like prototype which uses html tag library.

```

<html:html>
    <head>
        <title>DynaActionForm app</title>
    </title>
    <body>
        <html:form action="/product.do" >

        <html:base /><br/>

        <font color="red"><html:errors property="pid"/></font>
        Pid: <html:text property="pid" /><br/>

        <font color="red"><html:errors property="name"/></font>
        Name: <html:text property="name" /><br/>

        <font color="red"><html:errors property="price"/></font>
        Price: <html:text property="price" /><br/>
        <html:submit value="store" />
    </html:form>
    </body>
</html:html>

```

productForm.jsp

Step5:Develop an action class which is responsible to capture the data and store it database server.

The struts internal code is responsible to create FormBean object and capture the data and store it in FormBean object. This FormBean object supplied to parameter to execute() method.

In the execute() method from the actionform object we get the data and store in database server.

```

import java.sql.*;
import javax.servlet.http.*;
import org.apache.struts.action.*;

public class ProductAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form, HttpServletRequest request,
                                HttpServletResponse response) throws Exception{
        ProductFB objProductFB = (ProductFB) form;

        String pid = (String) objProductFB.get("pid");
        String name = (String) objProductFB.get("name");
        String price = (String) objProductFB.get("price");

        //storing the data in db
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        Connection objConnection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "bms", "abc");
        String insertQuery = "insert into product values(?, ?, ?)" ;
        PreparedStatement PreparedStatement = objConnection.prepareStatement(insertQuery);
        PreparedStatement.setString(1,pid);
        PreparedStatement.setString(2,name);
        PreparedStatement.setString(3,price);

        PreparedStatement.executeUpdate();

        //storing the data request scope
        request.setAttribute("pid", pid);
        request.setAttribute("name", name);
        request.setAttribute("price", price);

        return mapping.findForward("stored");
    }
}

```

ProductAction.java

Step6:Configure the action class in struts configuration file

```

<action-mappings>
    <action path="/product" type="ProductAction" name="productDetailsForm"
        validate="true" scope="request" input="/productForm.jsp">
        <forward name="stored" path="/success.jsp" />
    </action>
</action-mappings>

```

Step 7: develop a success form (success view).

This is responsible to display success message to the user.

```

<%@ page isELIgnored="false" %>

successfully stored the following details in db
pid: ${pid}<br/>
name: ${name}<br/>
price: ${price}<br/>

<a href="/productForm.jsp">click here to add another record </a>

```

success.jsp

Deploy the project and test it. Once we have deployed the above project we have to start testing the project from the jsp(productForm.jsp).

Plugins:

Plugin is a piece of software. We can integrate plugin software with any other software. Once if you integrate a plugin software with any other software the capability of existing software will be increased.

By using IDE's like eclipse we can't perform the operations like developing AWT and Swing application quickly. If we want eclipse IDE to carryout AWT and swings we need to download a plugin and add the plugin to eclipse software.

We can use a plugin visual editor to develop AWT and swing applications in eclipse IDE. The advantage of this approach is we can finish the work quickly when we compared with manual work.

To work with hibernate in eclipse IDE we have to integrate JBossTools Plugin with eclipse ide.

There are multiple ways are available to integrate the plugins with eclipse ide.

1. Download the plugin, Extract the contents of plugin and place them in eclipse folder.
2. We can directly install the plugin in eclipse software .

Help---→ Install New software -----→ add ----→ Provide plugin name and url and click on ok.

1. Know your Eclipse & JBoss Tools version to download

First, you have to find out the correct version of Hibernate/JBoss tool for your Eclipse IDE. Go here – <http://www.jboss.org/tools/download> for the available combination version to download.

For example,

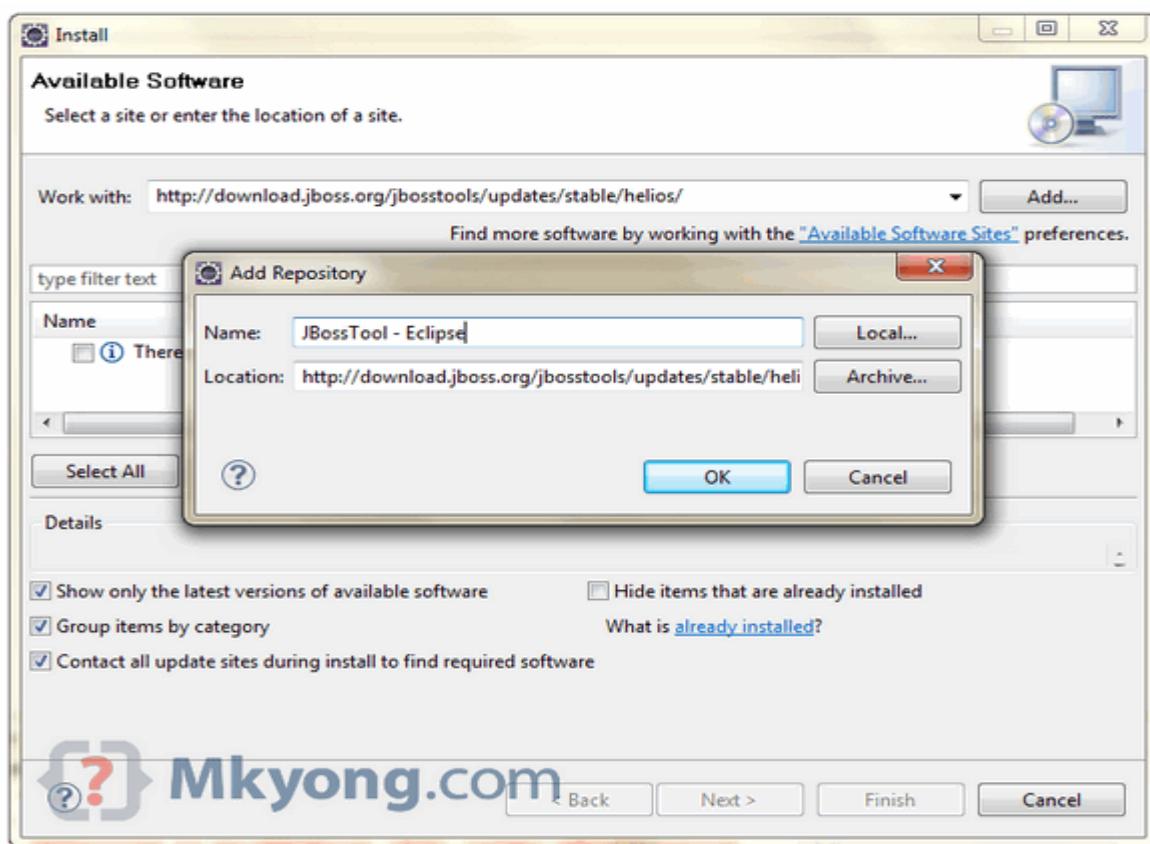
1. If you are using Eclipse 3.6 / Helios , download JBoss Tools 3.2
2. If you are using Eclipse 3.5 / Galileo, download JBoss Tools 3.1

2. Eclipse update site for JBoss Tools

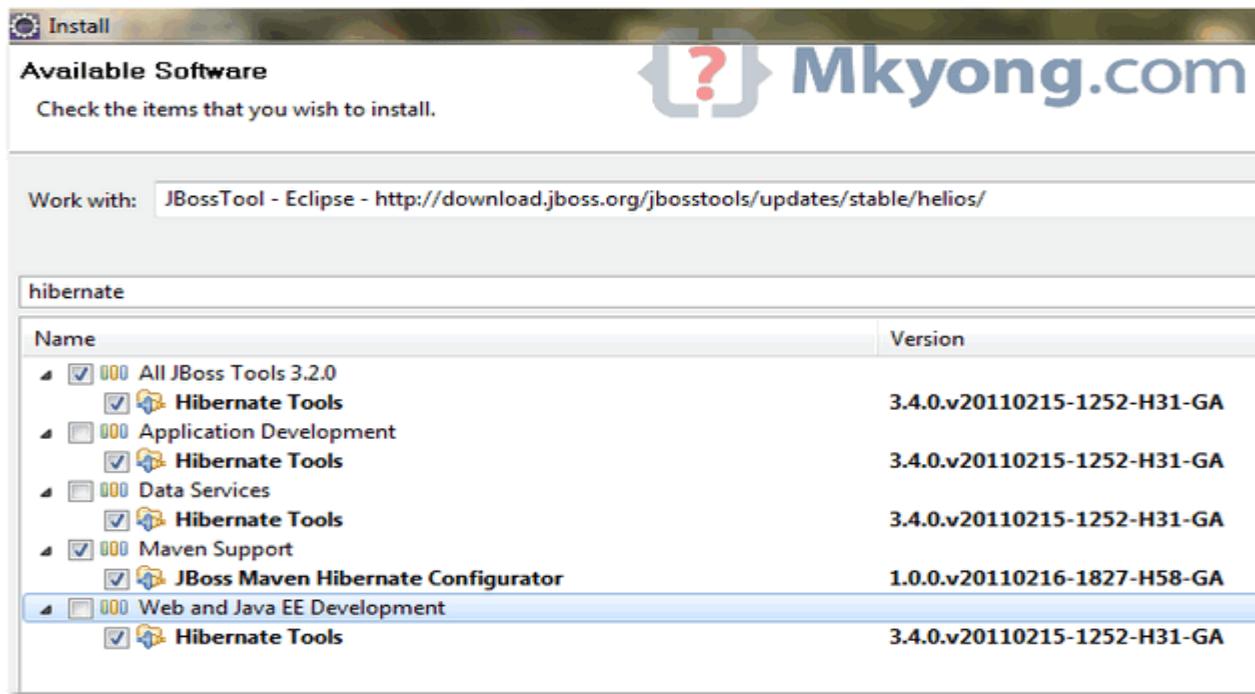
Point to your correct version, right click to copy the Eclipse update site for JBoss tools. For Eclipse 3.6, the URL is "<http://download.jboss.org/jbosstools/updates/stable/helios/>"

3. Install It

In Eclipse IDE, menu bar, select “Help” >> “Install New Software ...” , put the Eclipse update site URL.



Type “hibernate” in the filter box, to list down the necessary components for Hibernate tools. Select all the “Hibernate Tools” components and click next to download.



Warning

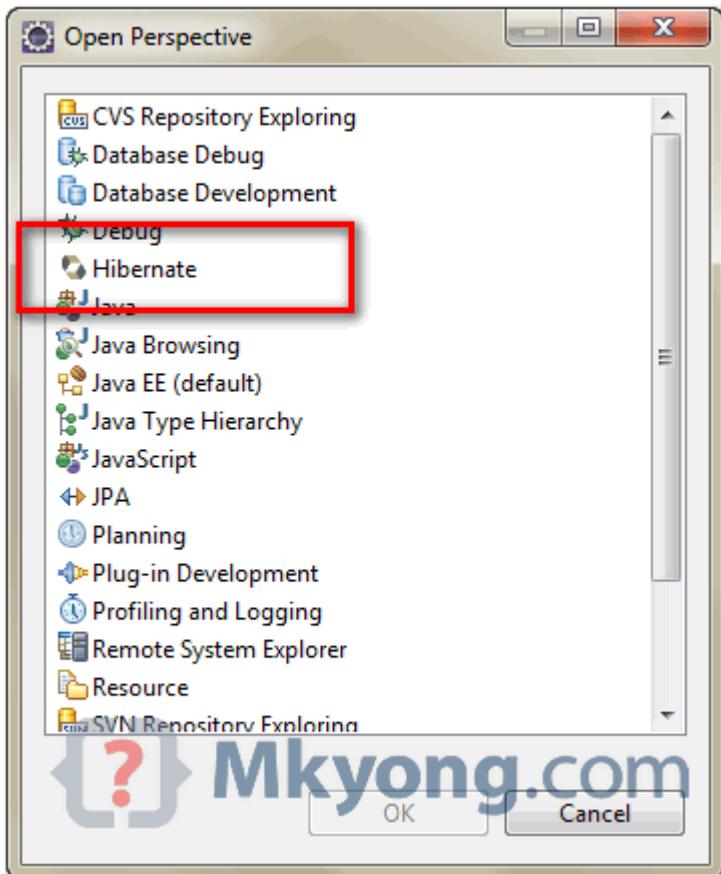
Do not select all components, it will take much longer time download many unnecessary components. You want Hibernate tools only, not others.

4. Restart Eclipse

After the download progress is completed, restart Eclipse to take effect.

5. Verification

If Hibernate tools is installed properly, you are able to see the “Hibernate Perspective” in “Windows” >> “Open Perspective” >> “Others”.



Done.

In Struts we can develop our own plugins and we can integrate.

Procedure to use plugins struts:

1. Develop a plugin class
2. Configure a plugin class into struts configuration file

What is plugin class?

A class which provides the implementation of plugin interface is called as Plugin class.

Developing a plugin class:

```
public class TestPlugin implements Plugin {  
    public void init(ActionServlet servlet, ModuleConfig config){  
        System.out.println("Init() is called");  
    }  
    public void destroy(){  
        System.out.println("destroy() is called");  
    }  
}
```

2)configuring that in struts configuration file.

```
<struts-config>
    <plug-in className="TestPlugIn" />
</struts-config>
```

we can supply property to the <plug-in> tag . To do that in the plug-in class we have to supply a property and we need to supply the value to it.

```
//read the contents of configuration file
public class TestPlugIn implements PlugIn {
    String fileName;
    public void setFileName(String fileName){
        this.fileName= fileName;
    }
    public String getFileName(){
        return fileName;
    }
    public void init(ActionServlet servlet, ModuleConfig config){
        System.out.println("Reading the configuration from " + fileName);
        ServletContext application = servlet.getServletContext();
        application.setAttribute("fileName", fileName);
    }
    public void destroy(){
        fileName = null;
        System.out.println("removing contents from jvm memory");
    }
}
```

We have to configure the Plugin class in struts configuration file.

```
<plug-in className="TestPlugIn" >
    <set-property property="fileName" value="test.xml" />
</plug-in>
```

struts-config.xml

When struts 1.0 is released they have not taken care of performing validations. In struts 1.1 they have provided the code to take care of validations. But the Struts guys are released the software in the form of Plugin. The advantage of this we can integrate this software with any other software.

From Struts 1.1 onwards apache guys itself has integrating validator plugin and releasing Struts software we need to use validator plugin directly in our project.

Form based application using DynaValidatorForm :

we have to follow the following steps to develop form based apps:

When we are working with DynaValidatorForm we no need to write setter() and getter() methods in FormBean class. Internally it is implementing to DynaBean as part of this interface it is providing the setter and getter() methods. DynaBean is available in commons-beanutils.jar file.

Step1 :Get the Prototype. From the prototype identify the following three elements

7. Identify the field names (Property name) for every field available in that form.
8. Identify the form name (the name shouldn't contain any spaces)
9. Identify the all the validations required on that form.

eg:

The diagram illustrates a User Registration Form with the following fields and their corresponding DynaBean properties:

- User Name: → `userName`
- Password: → `password`
- Retype password: → `retypePassword`
- DOB: → `dob`
- Email Id: → `emailId`
- Age: → `age`

Below the input fields is a **Register** button. At the bottom of the form, the name is specified as `name: UserRegistrationForm`.

Validations to carry out the above form:

1. All fields are mandatory
2. UserName must be minimum of 5 characters
3. Password and retype password must be same
4. Dob must be date format
5. age must be numeric
6. age must be ≥ 21 and ≤ 30

Step2: Develop the FormBean class based on DynaValidatorForm

```
org.apache.struts.action.ActionForm  
  
void reset(ActionMapping ,HttpServletRequest)  
// Can be used to reset bean properties to their default  
state, as needed.  
  
ActionErrors validate(ActionMapping,HttpServletRequest)  
//Can be used to validate the properties
```



```
org.apache.struts.action.DynaActionForm  
  
set(propertyName, objectValue)  
get(propertyName)
```



```
org.apache.struts.validator.DynaValidatorForm  
  
//override the validate() to take care  
of validations by reading validation.xml  
file
```



```
UserRegistrationFB
```

```
.
```

Developing UserRegistrationFB class :

We no need to write any thing in UserRegistrationFB class. this is because setter and getter() methods are inherited from DynaBean and validate() inherited from DynaValidatorForm.

In DynaValidatorForm they have override a validate() to take care of validations.

```
public class UserRegistrationFB extends DynaValidatorForm {
```

```
}
```

Step3: configure the FormBean class in struts configuration file

1. copy the FormBean classes into classes folder.
2. configure the FormBean class in struts configuration file in <form-beans> tag

```

<form-beans>
    <form-bean name="userRegistrationForm" type="UserRegistrationFB">
        <form-property name="userName" type="java.lang.String"/>
        <form-property name="password" type="java.lang.String"/>
        <form-property name="retypePassword" type="java.lang.String"/>
        <form-property name="emailId" type="java.lang.String"/>
        <form-property name="age" type="java.lang.String"/>
        <form-property name="dob" type="java.lang.String"/>
    </form-bean>
</form-beans>

```

struts-config.xml

Step4:Develop the input form or form view

The following form responsible to display the form as like prototype which uses html tag library.

```

<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>

<html:html>
    <head>
        <title>User RegistrationForm using Dynavalidatorform</title>
    </head>
    <body>
        User Registration form
        <html:form action="/register">
            <html:base />
            <html:errors property="userName" /><br/>
            UserName: <html:text property="userName" /><br/>

            <html:errors property="password" />
            Password: <html:password property="password" /><br/>
            <html:errors property="retypePassword" />
            Retype Password: <html:password property="retypePassword" /><br/>
            <html:errors property="dob" />
            DOB : <html:text property="dob" /><br/>
            <html:errors property="emailId" />
            EmailId: <html:text property="emailId" /><br/>
            <html:errors property="age" />
            Age: <html:text property="age" /><br/>

            <html:submit value="Register" property="register" />
            <html:reset />
        </html:form>
    </body>
</html:html>

```

UserRegister.jsp

Step5:Develop an action class which is responsible to capture the data and store it database server.

The struts internal code is responsible to create FormBean object and capture the data and store it in FormBean object. This formbeanobject supplied to parameter to execute() method .

In the execute() method from the actionform object we get the data and store in database server.

```
public class UserRegistrationAction extends Action {  
  
    public ActionForward execute(ActionMapping mapping, ActionForm form,  
                                HttpServletRequest request, HttpServletResponse response) {  
  
        UserRegistrationFB objUserRegistrationFB = (UserRegistrationFB) form;  
  
        //Code to store the data in database server  
        return mapping.findForward("done");  
    }  
}
```

UserRegistrationAction.java

Step6:Configure the action class in struts configuration file

```
<action-mappings>  
    <action name="userRegistrationForm" path="/register"  
           type="UserRegistrationAction" scope="request" validate="true"  
           input="/UserRegister.jsp">  
        <forward name="done" path="/success.jsp" />  
    </action>  
</action-mappings>
```

struts-config.xml

Step7: develop a success form (success view).

This is responsible to display success message to the user.

```
Successfully stored the data in db<br>  
  
<a href="UserRegister.jsp" >click here to register again </a>
```

Success.jsp

Deploy the project and test it. Once we have deploy the above project we have to start testing the project from the jsp (UserRegister.jsp).

We have to configure the all validations in validation.xml file as follows

```

<formset>
    <form name="userRegistrationForm">
        <field property="userName" depends="required,minlength">
            <arg position="0" key="UserName" resource="false"/>
            <arg position="1" name="minlength" key="${var:minlength}" resource="false"/>
            <var>
                <var-name>minlength</var-name>
                <var-value>5</var-value>
            </var>
        </field>
        <field property="password" depends="required,minlength">
            <arg position="0" key="Password" resource="false"/>
            <arg position="1" name="minlength" key="${var:minlength}" resource="false"/>
            <var>
                <var-name>minlength</var-name>
                <var-value>5</var-value>
            </var>
        </field>
        <field property="retypePassword" depends="required,validwhen">
            <arg name="required" position="0" key="Retype Password" resource="false"/>
            <arg name="validwhen" key="errors.rpwd" resource="true"/>
            <var>
                <var-name>test</var-name>
                <var-value>(*this* == password)</var-value>
            </var>
        </field>
        <field property="dob" depends="required,date">
            <arg position="0" key="Date of Birth" resource="false"/>
        </field>
        <field property="emailId" depends="required,email">
            <arg position="0" key="EmailId" resource="false"/>
        </field>
    </form>
</formset>

```

```

<field property="age" depends="required,integer,intRange">
    <arg position="0" key="Age" resource="false"/>
    <arg position="1" key="${var:min}" resource="false" />
    <arg position="2" key="${var:max}" resource="false" />
    <var>
        <var-name>min</var-name>
        <var-value>21</var-value>
    </var>
    <var>
        <var-name>max</var-name>
        <var-value>30</var-value>
    </var>
</field>

</form>
</formset>

```

Validator Plugin takes care of some predefined validations only. These are the common validations which are available across multiple projects.

Instead of we develop the code to take care of validations struts guys developed the code to take care of validations. All these code is available inside **common-validator.jar**.

Apache guys have developed some predefined classes to take care of validations.

```

public class FieldCheck {
    boolean required.FieldName, FieldValue,...) {
        //code to check whether user has entered data or not
        //Code to take care Validations

    }
}

```

FieldCheck.java

Everyvalidator is mapped with an error key. These error keys are configured in default properties file (MessageResources.properties).

eg: required---errors.required

The following are some of the validators and corresponding error codes:-

email ----errors.email

minlength ----errors.minLength

maxlength ---errors.maxlength

date ---errors.date

All these validators are configured in validation-rules.xml. This xml file is available in struts-core.jar.

The following is an example of validation-rule.xml

```

<form-validation>

    <global>

        <validator name="required"
                    classname="org.apache.struts.validator.FieldChecks"
                    method="validateRequired"
                    methodParams="java.lang.Object,
                                org.apache.commons.validator.ValidatorAction,
                                org.apache.commons.validator.Field,
                                org.apache.struts.action.ActionMessages,
                                org.apache.commons.validator.Validator,
                                javax.servlet.http.HttpServletRequest"
                    msg="errors.required"/>
    </global>
</form-validation>

```

validation-rules.xml

we can find all these validators in the documentation.

To use predefined validations we must configure all the validations in **validation.xml** file.

When we submit the form it will execute validate() of DynaValidatorForm.

The internal code of () of DynaValidatorForm validate () read the contents from validation.xml and perform the validations.

<bean:message>

By using `<bean:message>` tag we can read the value of Configurable messages(Which have parameters in that message) from properties file.

e.g.: `errorsmaxlength={0}` can not be greater than {1} characters.

```
<bean:message key=" errors maxlenlength" />
```

when we run above program it will display the message as “**null** can not be greater than **null** characters.”

But we want to replace parameter with a value like Age and 5.

To do that we use an attribute **arg**.

```
<bean:message arg0="UserName" arg1="10" key="errorsmaxlength" />
```

when we run above program it will display the message as “**UserName can not be greater than 10 characters.**”

In the above example **arg0** is used to replace parameter value with **UserName** and **arg1** is used to replace parameter value with **10**.

When we configuring validations we have to check what is the errorcode return by that Validator.we need to get the Configurable message for that validator. If message contains parameter we have to supply the value to it by using <arg> tag forEg: we are using **required** validator. This validator will return the **errors.required** error key. So we need to get the message for **errors.required** key from the properties file that is **“{0} is required”**. This message contains a parameter {0} .So we have to supply the value to parameter {0} by using <arg> tag

```
<arg name="" position="0" key="" resource="" />
```

<arg>tag is taking four attributes as shown above tag.

1) position :

This attribute indicates to which parameter number we are supplying the values. If we are not supply these attribute by default it will takes the value as zero. Afterwards it will increment by one.

Eg: **<arg position="0" key="User Name" resource="false" />**
(or)

```
<arg key="User Name" resource="false" />
```

2) key :

This attribute is dependent on **resource** attribute. The key attribute value is supply as input to parameter.

3) resource :

when **resource="false"** the key attribute value is supplied as input to parameter.

eg: <arg key="User Name" resource="false" />

for above tag it will display as

User Name is required

when **resource="true"** It takes the key attribute value and check whether the key value is specified in properties file or not. If key is not configured in Properties file it returns a null value. If it is configured it get the value of it and display to the client.

eg: **<arg key="User Name" resource="true" />**

For above tag it will check in properties file for UserName key. If it is there it will get the value of that and display to the user.

User Name is required

otherwise it displays **null is required**.

4) name :

We can supply name attribute to **<arg>** tag. This name indicates to which validator the argument value has to be supplied.

```
<field property="userName" depends="required,minlength">
    <arg key="userName" resource="true" />
    <arg name="minlength" key="${var:minlength}" resource="false" />
    <var>
        <var-name>minlength</var-name>
        <var-value>4</var-value>
    </var>
</field>
```

validation.xml

According to the above configuration the first **<arg>** tag is used by required validator and second **<arg>** tag is used by minlength validator.

Note: If we don't specify the name attribute by default the **<arg>** is applicable to all validators.i.e above first **<arg>** tag is applicable to all validators

By default the error groupName is same as propertyName in DynaValidatorForm.we use these groupNames to display the error messages at specific location.

eg: **<html:errors property="userName" />**

If we supply a validator which is not available in predefined list it will throw an

Exception saying: validatorException : no validator found

Different validators uses different variables we can find the all variables in manual. We can configure the variables in validation.xml as shown below.

```

<field property="userName" depends="required,minlength">
    <arg key="userName" resource="true" />
    <arg name="minlength" key="${var:minlength}" resource="false" />
    <var>
        <var-name>minlength</var-name>
        <var-value>4</var-value>
    </var>
</field>

```

validation.xml

By default date validator uses a class `java.text.SimpleDateFormat`. Whenever the object is created to this class it uses month-date-year format. If we would like to change the format we have to supply the value based on requirement. To supply our own date format we are using a variable **datePattern**. If we want the date pattern which is used in xml file and datePattern entered by the user must be same, in this scenario we use another variable **datePatternStrict**. When we use **datePatternStrict** it will verify the no.of characters which we entered.

```

<field property="userName" depends="required,minlength">
    <arg key="userName" resource="true" />
    <arg name="minlength" key="4" resource="false" />
    <var>
        <var-name>minlength</var-name>
        <var-value>4</var-value>
    </var>
</field>

```

validation.xml

The disadvantage of above configuration is when ever customer ask us to change minlength we have to do in two places. Because this we get maintainance related problems. To resolve this can define constant in global as shown in below

```

<global>
    <constant>
        <constant-name>userMinLength</constant-name>
        <constant-value>5</constant-value>
    </constant>
</global>

```

validations.xml

To use the above constant variable in our program we use EL Expressions as shown below.

```

<field property="userName" depends="required,minlength">
    <arg key="userName" resource="true" />
    <arg name="minlength" key="${var:minlength}" resource="false" />
    <var>
        <var-name>minlength</var-name>
        <var-value>${userMinLength}</var-value>
    </var>
</field>

```

validation.xml

Standard Built In Validations

Validator ships with a set of pre-defined validators, as follows:

required - mandatory field validation. Has no variables.

```

<field property="name" depends="required">
    <arg position="0" key="customer.name"/>
</field>

```

validwhen - validator for checking one field against another.

minlength - validate input data isn't less than a specified minimum length. Requires a minlength variable.

```

<field property="name" depends="required,minlength">
    <arg position="0" key="customer.name"/>
    <arg position="1" name="minlength" key="${var:minlength}" resource="false"/>
    <var><var-name>minlength</var-name><var-value>3</var-value></var>
</field>

```

maxlength - validate input data doesn't exceed a specified maximum length. Requires a maxlength variable.

```

<field property="name" depends="required,maxlength">
    <arg position="0" key="customer.name"/>
    <arg position="1" name="maxlength" key="${var:maxlength}" resource="false"/>
    <var><var-name>maxlength</var-name><var-value>30</var-value></var>
</field>

```

mask - validate format according to a regular expression. Requires a mask variable to specify the regular expression. Since version 1.1, the regular expression must start with a ^ and end with a \$ (see example below).

```

<field property="name" depends="required,mask">
    <msg name="mask" key="registrationForm.lastname.maskmsg"/>
    <arg position="0" key="registration.name"/>
    <var><var-name>mask</var-name><var-value>^[a-zA-Z]*$</var-value></var>
</field>

```

byte - validates that a field can be converted to a Byte.

```
<field property="age" depends="byte">
<arg position="0" key="employee.age"/>
</field>
```

short - validates that a field can be converted to a Short.

```
<field property="productnumber" depends="short">
<arg position="0" key="order.prodno"/>
</field>
```

integer - validates that a field can be converted to an Integer.

```
<field property="ordernumber" depends="integer">
<arg position="0" key="order.number"/>
</field>
```

long - validates that a field can be converted to a Long.

```
<field property="ordernumber" depends="long">
<arg position="0" key="order.number"/>
</field>
```

float - validates that a field can be converted to a Float.

```
<field property="amount" depends="float">
<arg position="0" key="sale.amount"/>
</field>
```

double - validates that a field can be converted to a Double.

```
<field property="amount" depends="double">
<arg position="0" key="sale.amount"/>
</field>
```

date - validates that a field can be converted to a Date. This validator uses java.text.SimpleDateFormat to parse the date and optionally either a datePattern or datePatternStrict variable can be used. If no pattern is specified the default short date format is assumed. The difference between using the datePatternStrict and datePattern variables is that datePatternStrict checks additionally that the input data is the same length as the pattern specified (so for example 1/1/2004 would fail with a pattern of MM/dd/yyyy).

```
<field property="saledate" depends="required,date">
<arg position="0" key="myForm.saledate"/>
<var><var-name>datePattern</var-name><var-value>MM/dd/yyyy</var-value></var>
</field>

<field property="saledate" depends="required,date">
<arg position="0" key="sale.orderdate"/>
<var><var-name>datePatternStrict</var-name><var-value>MM/dd/yyyy</var-value></var>
</field>
```

range - validate number range.

Deprecated, use intRange, longRange, floatRange
ordoubleRange.

intRange - validates that an integer field is within a specified range. Requires min and max variables to specify the range. This validator depends on the integer validator which must also be in the field's depends attribute.

```
<field property="age" depends="required,integer,intRange">
<arg position="0" key="employee.age"/>
<arg position="1" name="intRange" key="${var:min}" resource="false"/>
<arg position="2" name="intRange" key="${var:max}" resource="false"/>
<var><var-name>min</var-name><var-value>18</var-value></var>
<var><var-name>max</var-name><var-value>65</var-value></var>
</field>
```

longRange - validates that a long field is within a specified range. Requires min and max variables to specify the range. This validator depends on the long validator which must also be in the field's depends attribute.

```
<field property="age" depends="required,long,longRange">
<arg position="0" key="employee.age"/>
<arg position="1" name="longRange" key="${var:min}" resource="false"/>
<arg position="2" name="longRange" key="${var:max}" resource="false"/>
<var><var-name>min</var-name><var-value>18</var-value></var>
<var><var-name>max</var-name><var-value>65</var-value></var>
</field>
```

floatRange - validates that a float field is within a specified range. Requires min and max variables to specify the range. This validator depends on the float validator which must also be in the field's depends attribute.

```
<field property="ordervalue" depends="required,float,floatRange">
<arg position="0" key="order.value"/>
<arg position="1" name="floatRange" key="${var:min}" resource="false"/>
<arg position="2" name="floatRange" key="${var:max}" resource="false"/>
<var><var-name>min</var-name><var-value>100</var-value></var>
<var><var-name>max</var-name><var-value>4.99</var-value></var>
</field>
```

doubleRange - validates that a double field is within a specified range. Requires min and max variables to specify the range. This validator depends on the double validator which must also be in the field's depends attribute.

```
<field property="ordervalue" depends="required,double,doubleRange">
<arg position="0" key="employee.age"/>
<arg position="1" name="doubleRange" key="${var:min}" resource="false"/>
<arg position="2" name="doubleRange" key="${var:max}" resource="false"/>
<var><var-name>min</var-name><var-value>100</var-value></var>
<var><var-name>max</var-name><var-value>4.99</var-value></var>
</field>
```

creditCard - validate credit card number format

```
<field property="name" depends="required,creditCard">
<arg position="0" key="customer.cardnumber"/>
```

```
</field>
```

email - validate email address format

```
<field property="customeremail" depends="email">
<arg position="0" key="customer.email"/>
</field>
```

url - validates url format. Has four *optional* variables (`allowallschemes` , `allow2slashes` , `nofragments` and `schemes`) which can be used to configure this validator.

allowallschemes specifies whether all schemes are allowed. Valid values are true or false (default is false). If this is set to true then the schemes variable is ignored.

allow2slashes specifies whether double '/' characters are allowed. Valid values are true or false (default is false).

nofragments specifies whether fragement are allowed. Valid values are true or false (default is false - i.e. fragments are allowed).

schemes - use to specify a comma separated list of valid schemes. If not specified then the defaults are used which are http ,https and ftp .

```
<field property="custUrl" depends="url">
<arg position="0" key="customer.url"/>
</field>

<field property="custUrl" depends="url">
<arg position="0" key="customer.url"/>
<var>
<var-name>nofragments</var-name>
<var-value>true</var-value>
</var>
<var>
<var-name>schemes</var-name>
<var-value>http,https,telnet,file</var-value>
</var>
</field>
```

Constants/Variables

Global constants can be inside the global tags and FormSet/Locale constants can be created in the formset tags. Constants are currently only replaced in the Field's property attribute, the Field's var element value attribute, the Field's msg element key attribute, and Field's arg element's key attribute. A Field's variables can also be substituted in the arg elements (ex: \${var:min}). The order of replacement is FormSet/Locale constants are replaced first, Global constants second, and for the arg elements variables are replaced last.

```
<global>
<constant>
<constant-name>zip</constant-name>
<constant-value>^\d{5}(-\d{4})?$_</constant-value>
</constant>
```

```
</global>
```

The var element under a field can be used to store variables for use by a pluggable validator. These variables are available through the Field's getVar(String key) method.

```
<fieldproperty="integer"depends="required,integer,intRange">
<arg position="0" key="typeForm.integer.displayname"/>
<arg position="1" name="intRange"key="${var:min}"resource="false"/>
<arg position="2" name="intRange"key="${var:max}"resource="false"/>
<var>
<var-name>min</var-name>
<var-value>10</var-value>
</var>
<var>
<var-name>max</var-name>
<var-value>20</var-value>
</var>
</field>
```

Designing Complex Validations with validwhen

[Since Struts 1.2.0] A frequent requirement in validation design is to validate one field against another (for example, if you have asked the user to type in a password twice for confirmation, to make sure that the values match.) In addition, there are fields in a form that may only be required if other fields have certain values. The validwhen validator is designed to handle these cases.

The validwhen validator takes a single var field, called test . The value of this var is a boolean expression which must be true in order for the validation to success. The values which are allowed in the expression are:

- The token *this* , which contains the value of the field currently being tested
- Other fields in the form referenced by field name, such as customerAge
- The token null which will match against either null or an empty string
- Single or double-quoted string literals.
- Integer literals in decimal, hex or octal format

Using our own validator.xml

Instead of using validator.xml file for validations we can use our own xml file as validator.xml. But we need to configure it in Struts configuration file.

Instead of using predefined validator_rules.xml file, we can use our own xml file and we need to configure it in struts configuration file. As shown below file we change the configuration in struts-config.xml

Here validation_test.xml file is our own validation file. Generally validator_rules.xml file is available in struts-core.jar. But here we placed it in WEB-INF folder.

```

<struts-config>
    ...
    <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
        <set-property property="pathnames"
                      value="/WEB-INF/validator-rules.xml,
                           /WEB-INF/validation_test.xml"
        </plug-in>
<struts-config>

```

struts-config.xml

Procedure to work with Custom Validators:

1. Develop a class with static method by supplying the standard parameters to a method.

```

import org.apache.commons.validator.Field;
import org.apache.commons.validator.Validator;
import org.apache.commons.validator.ValidatorAction;
import org.apache.struts.action.ActionMessages;

public class CustomValidator{
    public CustomValidator() {
        System.out.println("CustomValidator object created***");
    }
    public static boolean checkFields(Object object, ValidatorAction va, Field
paramField, ActionMessages paramActionMessages, Validator paramValidator,
HttpServletRequest paramHttpServletRequest) {
        System.out.println("****Bean object name" + paramObject.getClass());

        String str = paramField.getProperty();

        UserRegistrationFB urfb = (UserRegistrationFB)object;
        System.out.println(urfb.get(str) + "*****##");
        return false;
    }
}

```

CustomValidator

Creating Pluggable Validators

The methodParams attribute takes a comma separated list of class names. The method attribute needs to have a signature complying with the above list. The list can be comprised of any combination of the following:

- java.lang.Object - Bean validation is being performed on.
- org.apache.commons.validator.ValidatorAction - The current ValidatorAction being performed.
- org.apache.commons.validator.Field - Field object being validated.
- org.apache.struts.action.ActionMessages - The errors object to add an ActionMessage to if the validation fails.
- javax.servlet.http.HttpServletRequest - Current request object.
- javax.servlet.ServletContext - The application's ServletContext.

- org.apache.commons.validator.Validator - The current org.apache.commons.validator.Validator instance.
- java.util.Locale - The Locale of the current user.

2) Copy the .class file into classes folder. **Configure CustomValidator class in validator-rules.xml by specifying a validator key(short name for this class)**

```
<form-validation>
  <global>
    <validator name="checkField">
      classname="CustomValidator"
      method="checkFields"
      methodParams="java.lang.Object,
                    org.apache.commons.validator.ValidatorAction,
                    org.apache.commons.validator.Field,
                    org.apache.struts.action.ActionMessages,
                    org.apache.commons.validator.Validator,
                    javax.servlet.http.HttpServletRequest"
      msg="errors.check"/>
  </global>
</form-validation>
```

validation-rules.xml

3) Configure this validator-rules.xml file in struts-config xml file as shown below. Make sure that validator-rules .xml file is available in WEB-INF folder.

```
<struts-config>
  ...
  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property property="pathnames"
                  value="/WEB-INF/validator-rules.xml,
                        /WEB-INF/validation_test.xml">
  </plug-in>
</struts-config>
```

struts-config.xml

4) Configure our own validator class in validator_test.xml file

```
<formset>
  <form name="userRegistrationForm">
    <field property="userName" depends="required, checkField">
      <arg key="userName" resource="true" />
    </field>
  </form>
</formset>
```

our own validator
which is configured in
validator-rules.xml

validation_test.xml

Q) When we develop CustomValidator classes?

Ans: If there is a filed which is used across all the screens of project and predefined validator is not available then we will use validator class.

Q) When we will override validate() method in the FormBean class?

Ans: If there is a validation which has to take care of only one field that validation is carried out in validate () method. Then only we will override validate () to carry out the validation.

eg: we have fields like password and retype password and we want check whether both values are same or not . If values are different then we have to display an error message on jsp.

When we **override validate()** method we need to make sure that we calls super class validate() method. This is because the super class validate() takes care of reading the contents from validation.xml and perform validations which are configured as part of validation.xml. If we are not call the validate () method then predefined validation's can't be carried out.

Eg:

```
public class UserRegistrationFB extends DynaValidatorForm {  
    public ActionErrors validate(ActionMapping mapping, HttpServletRequest  
        request) {  
        String password = (String) get("password");  
        String retypePassword = (String) get("retypePassword");  
        ActionErrors actionErrors = super.validate(mapping, request);  
  
        if(!password.equals(retypePassword)) {  
            actionErrors.add("password", new ActionMessage("errors.passwordMismatch"));  
        }  
  
        return actionErrors;  
    }  
}
```

UserRegistrationFB.java

Generating JavaScript using DynaValidatorForm:

By using **DynaValidatorForm**we can generate client side validations also.

To generate javascript in jsp we use the **<html:javascript />**

the mandatory attribute is **formName**. For formName attribute we have to supply form name as input to that.

Eg: **<html:javascriptformName="UserRegistrationForm" />**

when the above tag is evaluated it will generate a javaScript function whose name

validate+formName attribute value.eg: Here it will generate

validateUserRegistrationForm(form) in userRegistration.jsp

By default the **DynaValidatorForm**will not call the

validateUserRegistrationForm(form)method. We have to call that method explicitly. We are using onsubmit event to call the above JavaScript function.as shown below.

```

<html:javascript formName="userRegistrationForm" />

<html:form action="/register"
    onsubmit="return validateUserRegistrationForm(this)">
    //code to display the form using struts html tag library
</html:form>

```

UserRegister.jsp

The DynaValidatorForm plugin can generate JavaScript only for predefined validations. If we want to carry out our own validations by using JavaScript we have to write our own JavaScript functions.

Apache guys has developed their own validations using JavaScript and placed inside validator plugin.jar file. The validator plugin take the JavaScript form this jar file and place the JavaScript in jsp whenever we use **<html:javascript />**

Whenever we use our own JavaScript function to carry out client side validations from that also we can call the generated JavaScript function as shown below.

```

<script>
    function validate(form) {
        var status = validateUserRegistrationForm(form) ;
        return status;
    }
</script>

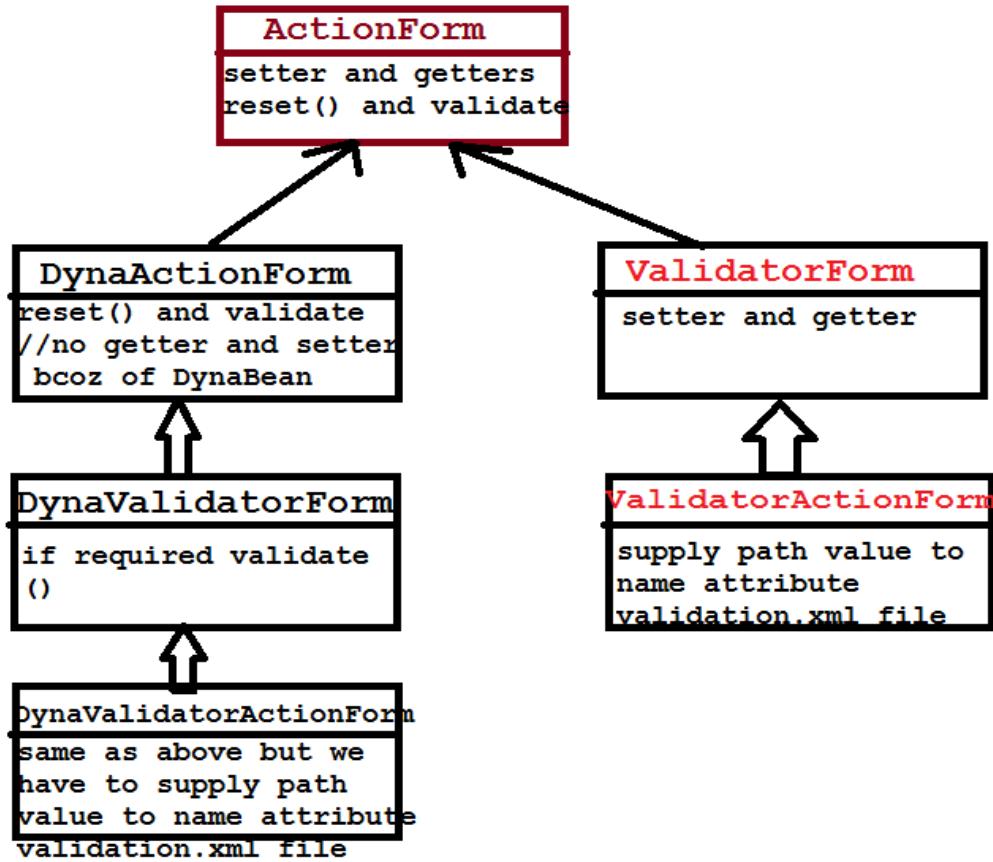
<html:javascript formName="userRegistrationForm" />
<html:form action="/register" onsubmit="return validate(this)">
    //Form fields.....
</html:form>

```

UserRegister.jsp

There are n no. of ways are there to develop FormBean classes. Some of them are

1. ActionForm
2. DynaActionForm
3. DynaValidatorForm
4. DynaValidatorActionsForm
5. ValidatorForm
6. ValidatorActionForm



Hierachy of FormBean classes

We can use other ways of using FormBean classes like **BeanValidatorForm**, **LazyValidatorForm** and etc.

Form based application using ValidatorForm :

When we working with **validatorForm** we no need to take care of the validations. We have to provide `setter()` and `getter` methods for identified properties. e.g: public class **SignupFB** extends **ValidatorForm** {

```

String firstName;
String lastName;
//provide setter and getters for above properties
  }
```

The remaining procedure is same as above **DynaValidatorForm**.

Form based application using ValidatorActionForm :

When we working with **validatorActionForm** we no need to take care of the validations. We have to provide `setter()` and `getter` methods for identified properties.

```
public class SignupFB extends ValidatorActionForm {
```

```

String firstName;
String lastName;
//provide setter and getters for above properties
  }
```

To compile above program we have to set the classpath to **struts-extras.jar**.

When we develop an application based on **ValidatorActionForm**, we have to configure the **validation.xml** as follows.

```

<struts-config>
<action-mappings>
    <action path="/signUp" type="com.desconit.struts.SignUpAction"
        name="signUpForm" scope="request" input="/SignUpForm.jsp" >
        <forward name="home" path="/home.jsp" />
    </action>
</struts-config>

```

struts-config.xml

```

<formset>
    <form name="/signUp"> path attribute value specified for action
        <field depends="required" property="firstName">
            <arg key="First Name" resource="false"/>
        </field>
    </formset>
</form-validation>

```

Validation.xml

In <form> tag for name attribute we have to supply the <action> tag path attribute specified value. Otherwise validations will not carryout.

In DynaValidatorActionForm also we should specify the path of the resource to name attribute in validator.xml as shown as above example.

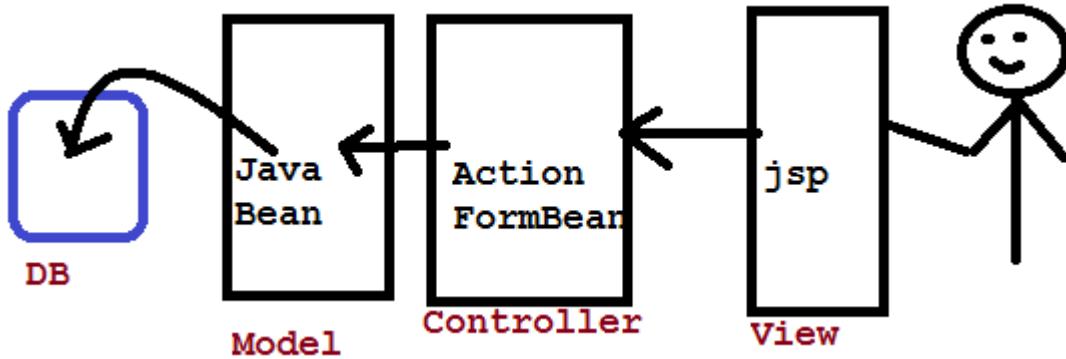
In Struts ActionServlet, Action classes, FormBean classes acts as Controllers. It's not recommended to write business logic in Action classes or FormBean classes. So we should not transfer the FormBean class to Model class.

Develop a complete Form Based application by using MVC2 architecture:

Your First Name:	<input type="text"/>	fieldnames firstName
Your Last Name:	<input type="text"/>	lastName
Your Email:	<input type="text"/>	emailId
Re-enter Email:	<input type="text"/>	retypeEmailId
New Password:	<input type="password"/>	password
I am:	<input type="radio"/> Male <input type="radio"/> Female	gender
Birthday:	<input type="text"/> 19 <input type="text"/> Aug <input type="text"/> 1989	birthday
<input type="button" value="SignUp"/>		

signUpForm

We have to capture the data from the above form and save that details in database server. The following is flow of the Project.



To implement the above requirement we have to perform the following steps:-

Step1:- Develop a javaBeanto write the business logic . The business logic is get the form details and store those in DataBase server and send an email to register mail id.
eg:

```

public class UserRegistrationJB {
    public boolean registerUser(String firstName, String lastName,
        String emailId, String password, String gender, String birthday) {
        boolean isRegistered = false;
        try {
            DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
            Connection connection = DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe", "bms", "abc");
            String insertUserRegistrationQuery = "insert into signupform values(?,?,?,?,?,?)";
            PreparedStatement objPreparedStatement = connection
                .prepareStatement(insertUserRegistrationQuery);
            objPreparedStatement.setString(1, firstName);
            objPreparedStatement.setString(2, lastName);
            objPreparedStatement.setString(3, emailId);
            objPreparedStatement.setString(4, password);
            objPreparedStatement.setString(5, gender);
            objPreparedStatement.setString(6, birthday);
            objPreparedStatement.executeUpdate();
            isRegistered = true;
            System.out.println("email has sent to "+ emailId);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

Step2:-Develop a FormBean class and configure that in Struts Configuration file.

In this program we don't require to override the reset() and validate() methods.that's why we are not writing any FormBean class but we are using predefined class DyanValidatorForm. configure that in struts-config.xml file

```

<form-beans>
    <form-bean name="signUpForm"
        type="org.apache.struts.validator.DynaValidatorForm">
        <form-property name="firstName" type="java.lang.String" />
        <form-property name="lastName" type="java.lang.String" />
        <form-property name="emailId" type="java.lang.String" />
        <form-property name="retypeEmailId" type="java.lang.String" />
        <form-property name="password" type="java.lang.String" />
        <form-property name="gender" type="java.lang.String" />
        <form-property name="birthday_day" type="java.lang.String" />
        <form-property name="birthday_month" type="java.lang.String" />
        <form-property name="birthday_year" type="java.lang.String" />
    </form-bean>
</form-beans>

```

struts-config.xml

Step3:-Configure the validation in validations.xml file as shown below

```

<formset>
    <form name="signUpForm">
        <field depends="required" property="firstName">
            <arg key="First Name" resource="false"/>
        </field>
        <field depends="required" property="lastName">
            <arg key="Last Name" resource="false"/>
        </field>
        <field depends="required" property="emailId">
            <arg key="Email Id" resource="false"/>
        </field>
        <field depends="required" property="retypeEmailId">
            <arg key="Retype EmailId" resource="false"/>
        </field>
        <field depends="required" property="password">
            <arg key="Password" resource="false"/>
        </field>
        <field depends="required" property="gender">
            <arg key="Gender " resource="false"/>
        </field>
    </form>
</formset>

```

validations.xml

Step4:-Develop an action to capture the data and call the methods of javaBean and forward the request to Welcome.jsp

```

public class SignUpAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        DynaValidatorForm objDynaValidatorForm = (DynaValidatorForm) form;
        //Getting the values from formbean object
        String firstName = (String) objDynaValidatorForm.get("firstName");
        String lastName = (String) objDynaValidatorForm.get("lastName");
        String emailId=(String)objDynaValidatorForm.get("emailId");
        String retypeEmailId=(String)objDynaValidatorForm.get("retypeEmailId");
        String password=(String)objDynaValidatorForm.get("password");
        String gender=(String)objDynaValidatorForm.get("gender");
        String birthday_day=(String)objDynaValidatorForm.get("birthday_day");
        String birthday_month=(String)objDynaValidatorForm.get("birthday_month");
        String birthday_year=(String)objDynaValidatorForm.get("birthday_year");
        String birthday = birthday_day+"-"+birthday_month+"-"+birthday_year;

        //create the object to javabean and call the registerUser()
        UserRegistrationJB objUserRegistrationJB = new UserRegistrationJB();
        boolean isRegistered = objUserRegistrationJB.registerUser(firstName, lastName,
            emailId, password, gender, birthday);

        request.setAttribute("firstName", firstName);
        return mapping.findForward("home");
    }
}

```

SignUpAction.java

Configure the above classes in Struts configuration file

```

<action-mappings>
    <action path="/signUp" type="com.desconit.struts.SignUpAction"
        name="signUpForm" scope="request" input="/SignUpForm.jsp"
        validate="true">
        <forward name="home" path="/home.jsp" />
        <forward name="failed" path="/SignUpForm.jsp" />
    </action>
</action-mappings>

```

Struts-config.xml

In the above project the java bean program is clubbed with business logic and Data Access Logic. It's always recommended to separate out the business logic and Data Access Logic. To separate out the business logic and Data AccessLogic we use DAO designPattern.

DAO DesignPattern:

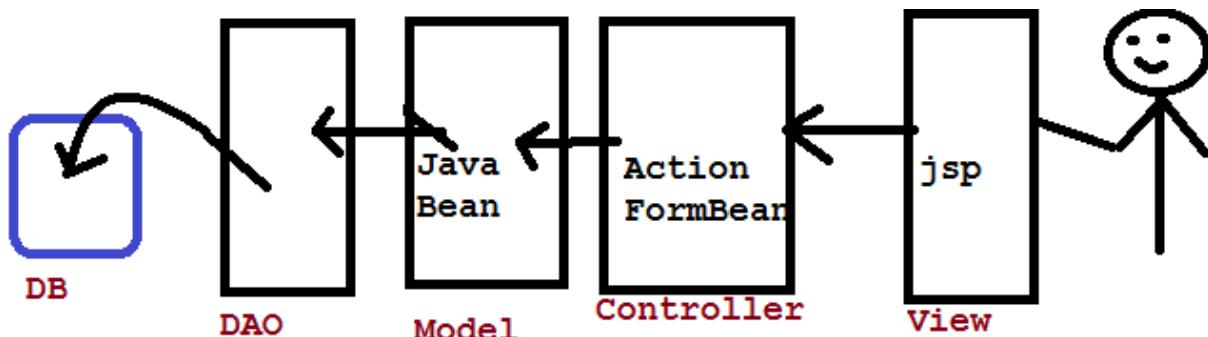
Your First Name:	<input type="text"/>	fieldnames firstName
Your Last Name:	<input type="text"/>	lastName
Your Email:	<input type="text"/>	emailId
Re-enter Email:	<input type="text"/>	retypemailId
New Password:	<input type="text"/>	password
I am:	Male <input type="button" value="▼"/>	gender
Birthday:	<input type="text"/> 19 <input type="text"/> Aug <input type="text"/> 1989	birthday
<input type="button" value="SignUp"/>		

signUpForm

DAO stands for **Data Access object**. DAO design pattern make sure that we clearly separate out business logic and Data Access Logic.

In the DAO classes we provide the code to interact with the data base server only. In this we will use jdbc or hibernate or any persistent technologies. So if any change in Persistent code we will modify only Dao classes. It will not affect on any other layer.

Develop the following formbased application using mvc2 and DAO,DTO design pattern



To implement the above project we have to perform the following steps:

DTO DesignPattern:

It resolves the problems of transferring the data from one component to another component. In the above example we transfer the data like **signup form** details in the form method parameters to JavaBeans from Controller. This is not recommended approach because when ever any change in the parameter we have to modify our program because of this we may face maintenance related problems. Instead of passing each value as method parameter to javabean we will create a DTO which contains instance variable and setter and getter methods and we will supply this DTO as

parameter to javabean. And it will call the DAO method which takes the DTO as parameter. DTO will pass from Controller to DAO

Develop a project which uses MVC2 designpattern DAO and DTO designpatterns.

step1:- Create a DTO class which can hold the FormBean data. This class has store the data of which enter by the user in the above shown form

```
public class UserRegistrationDTO implements Serializable {  
    String firstName;  
    String lastName;  
    String emailId;  
    String retypeEmailId;  
    String password;  
    String gender;  
    String birthday;  
  
    public String getFirstName() {  
        return firstName;  
    }  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
    //setter and getter for all methods  
}
```

UserRegistrationDto.java

step2:- Develop a DAO program. In this we write the only the code which is responsible to interact with the database server. This DAO class component takes the DTO object as parameter as shown below. This method is called by the Controller as it shown in controller program.eg:

```
public class UserRegistrationDAO {  
    public boolean registerUser(UserRegistrationDTO urDTO) {  
        boolean isRegistered = false;  
        //get the connection  
        String query = "insert into signupform values(?,?,?,?,?,?)";  
        PreparedStatement pstmt = connection  
            .prepareStatement(query);  
        pstmt.setString(1, objUserRegistrationDTO.getFirstName());  
        pstmt.setString(2, objUserRegistrationDTO.getLastName());  
        pstmt.setString(3, objUserRegistrationDTO.getEmailId());  
        pstmt.setString(4, objUserRegistrationDTO.getPassword());  
        pstmt.setString(5, objUserRegistrationDTO.getGender());  
        pstmt.setString(6, objUserRegistrationDTO.getBirthday());  
  
        pstmt.executeUpdate();  
        isRegistered = true;  
        return isRegistered;  
    }  
}
```

UserRegistrationDAO

In above example we have a registerUser() method which is responsible to store the user details in db. This method is returning Boolean value. If the data is successfully

registered then this method returns true otherwise it returns false. We used this value in controller to forward the appropriate jsp.eg. If user data is successfully registered then forward to success.jsp otherwise forward to inputform.jsp

step3:-Develop a JavaBean which is responsible to get the data from Controller, create an object to DAO class and call the methods of Dao. Usually in the JavaBean we will write the business logic code.

```
public class UserRegistrationJB {  
    public boolean registerUser(UserRegistrationDTO urDTO) {  
        boolean isRegistered = false;  
  
        UserRegistrationDAO urDAO = new UserRegistrationDAO();  
        isRegistered = objUserRegistrationDAO  
            .registerUser(objUserRegistrationDTO);  
  
        if (isRegistered) {  
            System.out.println("Mail has been sent to "  
                + objUserRegistrationDTO.getEmailId());  
        }  
        return isRegistered;  
    }  
}
```

UserRegistrationJB.java

Here We are calling the registerUser() method which is returning a value. we storing that in isRegistered variable . If user details are stored then send a mail to that email.

step4:-Develop FormBean class and Configure in struts configuration file.

In this example we no **need to display default values** to the form. That's why we are not overriding the reset() method. And we no need to take care the user defined validation that means predefined validations are enough to carry out the validations. So in this scenario we no need to develop our own FormBean class for that we will use predefined FormBean class. In this example we are using DynaValidatorForm.

The following is configurations of DynaValidatorForm in struts configuration file.

```

<struts-config>
<form-beans>
    <form-bean name="signUpForm"
        type="org.apache.struts.validator.DynaValidatorForm">
        <form-property name="firstName" type="java.lang.String" />
        <form-property name="lastName" type="java.lang.String" />
        <form-property name="emailId" type="java.lang.String" />
        <form-property name="retypeEmailId" type="java.lang.String" />
        <form-property name="password" type="java.lang.String" />
        <form-property name="gender" type="java.lang.String" />
        <form-property name="birthday_day" type="java.lang.String" />
        <form-property name="birthday_month" type="java.lang.String" />
        <form-property name="birthday_year" type="java.lang.String" />
    </form-bean>
</form-beans>

```

Struts-config.xml

step5:-Develop Action class and capture the data and create object to Javabean and call the methods of Javabean as follow

```

public class SignUpAction extends Action {
    public ActionForward execute(...) {
        DynaValidatorForm dynaform = (DynaValidatorForm) form;
        UserRegistrationDTO urDTO = new UserRegistrationDTO();
        urDTO.setFirstName((String)dynaform .get("firstName"));
        urDTO.setLastName((String) dynaform .get("lastName"));
        urDTO.setEmailId((String) dynaform .get("emailId"));
        urDTO.setPassword((String) dynaform .get("password"));
        urDTO.setGender((String) dynaform .get("gender"));
        objUserRegistrationDTO.setBirthday(birthday);

        // create the object to javabean and call the registerUser()
        UserRegistrationJB urJB = new UserRegistrationJB();
        boolean isRegistered = urJB .registerUser(urDTO);
        if (isRegistered) {
            request.setAttribute("firstName",dynaform .get("firstName"));
            return mapping.findForward("home");
        } else {
            ActionErrors objActionErrors = new ActionErrors();
            objActionErrors.add("signupError",
                new ActionMessage("signupfailed"));
            saveErrors(request, objActionErrors);
            return mapping.findForward("failed");
        }
    }
}

```

SignUpAction.java

If user data is successfully registered then forward to success.jsp otherwise forward to inputform.jsp . In Action class we can handle the errors. The errors in the Action class should be based on the return type of DAO and javabean methods.

step6:-Develop an input form and successform.

```

<html:form action="/signUp.do" onsubmit="return validateSignUpForm(this)">
    <html:errors/>
    <html:javascript formName="signUpForm"/>
    Your First Name:<html:text property="firstName" />
    Your Last Name:<html:text property="lastName" />
    Your Email:<html:text property="emailId" />
    Re-enter Email:<html:text property="retypeEmailId" />
    New Password:<html:password property="password" />
    I am:<html:select property="gender">
        <html:option value="male">Male</html:option>
        <html:option value="female">Female</html:option>
    </html:select>
    Birthday:<html:select property="birthday_day">
        <html:option value="1">1</html:option>
        <html:select property="birthday_month">
            <html:option value="Jan">Jan</html:option>
            <html:select property="birthday_year">
                <html:option value="1989">1989</html:option>
            <html:submit value="SignUp"></html:submit>
            <html:reset></html:reset>
        </html:select>
    </html:select>
</html:form>

```

SignupForm.jsp

VO Design Pattern: (Value Object)

VO design pattern is used retrieve the data from the DAO layer and send the data to view View Component. this design pattern is used to get the values from database server and store those values in Vo object. and transfer these vo object to View component by adding it to ArrayList.

eg :searchRecords:

Develop the following form based application to search for the records and display output to the client.

eg: searchRecords

Pid:	<input type="text"/>
Submit	

The business
in the pid field

logic is if user is entered any data
then it has to display the

corresponding record of product table from the database. If records are not available then display "No records found". If user doesn't enter any data, view has to display the all records of Product table.

we use the design patterns like **MVC2, DAO, DTO, VO** design patterns.

Step1: - Develop the DTO class. which is responsible to hold the pid field details. And transfer this dto to javabean from Controller.

ex:

```

public class SearchProductDTO {
    String pid;
    public String getPid() {
        return pid;
    }
    public void setPid(String pid) {
        this.pid = pid;
    }
}

```

SearchProductDTO.java

Step2:- Develop the VO class. This class responsibility is hold the product table data. It contains instance variables and setter and getter methods. These instance members based on the no of columns values we want store in VO object. This class is instantiated by DAO class and represent each record in the form of an VO object.

```

public class SearchProductVO {
    private String pid;
    private String name;
    private String price;

    public String getPid() {
        return pid;
    }
    public void setPid(String pid) {
        this.pid = pid;
    }
    //setter and getter for remaining fields
}

```

SearchProductVO.java

eg:

In this example we dealing with product table which contains only three columns that's why we are taking three fields in our VO class.

Step3: - Develop a DAO class which is responsible to represent the records in the form Objects and add these to ArrayList object.

```

public class ProductDAO {
    public ArrayList searchRecords(SearchProductDTO spDTO) {
        ArrayList list = new ArrayList();
        try {
            DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
            conn = DriverManager.getConnection("url", "bms", "abc");
            Statement stmt = conn.createStatement();

            //Forming the select query dynamically
            String selectQuery = "select * from product";

            if(spDTO.getPid() != null && !spDTO.getPid().equals("")) {
                selectQuery += " where pid='"+objSearchProductDTO.getPid()+"'";
            }

            ResultSet rsData = stmt.executeQuery(selectQuery);
            while(rsData.next()){
                SearchProductVO spVO = new SearchProductVO();
                spVO.setPid(rsData.getString("pid"));
                spVO.setName(rsData.getString("name"));
                spVO.setPrice(rsData.getString("price"));

                list.add(objSearchProductVO);
            }
        }catch(Exception exception){
            exception.printStackTrace();
        }
        return list;
    }//closing the method
}

```

SearchProductDAO.java

In

above program we forming the query dynamically.represent each record in the form SearchProductVO object. and store these objects to ArrayList. **searchRecords()** method is taking the DTO as parameter to it.and returning the ArrayList object.

Step4:- Develop the javabean (Model)class. It has to call the Dao methods .

```

public class ProductJB {
    public ArrayList searchRecords(SearchProductDTO spDTO) {
        ProductDAO objProductDAO = new ProductDAO();
        ArrayList list = objProductDAO.searchRecords(spDTO);

        return list;
    }
}

```

ProductJB

Model class responsibility is create the object to javabean and call the searchRecords(). return this arraylist object to controller. If we want to write any other business logic we will write in this class only.

Step5:- Develop the formbean class and configure that in struts configuration file. In this example we are not developing any formbean class just we are configuring the predefined Formbean class as shown below

```

<form-beans>
    <form-bean name="searchProductForm" type="DynaValidatorForm">
        form-property name="pid" type="java.lang.String" />
    </form-bean>
</form-beans>

```

struts-config.xml

Step6:- Develop the Action class and configure that in struts configuration. The action class responsibility is capture the data and stored it DTO object and call the javabeanmethods and get the arraylist object and store it in request scope.

```

public class SearchProductAction extends Action {
    public ActionForward execute(ActionMapping mapping,
        ActionForm form,HttpServletRequest req,HttpServletResponse res) {
        DynaValidatorForm dynaForm = (DynaValidatorForm) form;
        SearchProductDTO spDTO = new SearchProductDTO();
        spDTO.setPid((String)dynaForm.getString("pid"));
        ProductJB objProductJB = new ProductJB();
        ArrayList list = objProductJB.searchRecords(spDTO);

        request.setAttribute("productList", list);

        return mapping.findForward("display");
    }
}

```

SearchProductAction

Step6:- Develop the Input form(search.jsp)

Step7:-Develop the success form (display.jsp) which is responsible to get the arrayList object from that display the product details to user.

```

<body>
    <c:if test="${!empty productList}">
        <c:forEach var="SearchProductVO" items="${productList}">
            ${SearchProductVO.pid}
            ${SearchProductVO.name}
            ${SearchProductVO.price}<br>
        </c:forEach>
    </c:if>

    <c:if test="${empty productList}">
        No records found..<br/>
    </c:if>
    <a href="SearchProduct.jsp">Search again</a>
</body>

```

display.jsp

Hibernate integration in Struts:

The following are some of ways to integrate hibernate in struts:

1. write hibernate code in Action class (not recommended)
2. Based on Developing our own Hibernate plugin

3. By using SingleTonDesignpattern and hibernate Template design pattern

Procedure to use hibernate in struts:

1. copy all the hibernate related jar files into struts project lib folder.
2. coy hibernate configuration file and hbm (mapping) files into classes folder.

write hibernate code in Action class:

When we are writing hibernate code in Action class , for every request the action class execute() method will execute. so every time it is executing the configure() and buildSessionFactory() methods. But it is not recommended to call these methods every time. In a project life time we have to call these methods only once. Because these methods will take more amount of time to execute.

```
public class RetriveProductAction extends Action {  
    public ActionForward execute(ActionMapping mapping, ActionForm  
form,HttpServletRequest request, HttpServletResponse response) {  
  
    SessionFactory sf =  
        new Configuratin().configure().buildSessionFactory();  
  
    Session hsession = sf.openSession();  
    Query query = hsession.createQuery("from SearchProductVO");  
  
    ArrayList list = (ArrayList) query.list();  
  
    request.setAttribute("productList", list);  
    return mapping.findForward("display");  
}  
}  
  
RetriveProductAction
```

But it is not recommended to write the business logic in Controller and not recommended call hibernate configure() and buildSessionFactory() methods. For that we have to call these methods only once. This thing can be achieved by using plugins which will execute only once that is at time of deploying the project.

Based on Developing our own Hibernate plugin:

Procedure to use hibernate in struts:

1. copy all the hibernate related jar files into struts project lib folder.
2. coy hibernate configuration file and hbm (mapping) files into classes folder.
3. Develop a hibernate plugin class based on plugin interface
4. Configure the HibernatePlugin class in struts configuration file
`<plug-in className=" HibernatePlugin" />`
5. Develop an action class to get the SessionFactory object from the application scope and execute the query.

Develop a hibernate plugin class based on plugin interface

```

public class HibernatePlugIn implements PlugIn {
    public void destroy() {
    }

    public void init(ActionServlet servlet, ModuleConfig config)
        throws ServletException {
        SessionFactory sf = new Configuration().configure()
            .buildSessionFactory();
        System.out.println("got the sessionfactory-----");
        ServletContext application =servlet.getServletContext();
        application.setAttribute("sessionFactory", sf);
    }
}

```

HibernatePlugin

Develop an action class to get the SessionFactory

```

public class RetriveProductAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        SessionFactory sf = (SessionFactory)
            servlet.getServletContext().getAttribute("sessionFactory");
        Session hsession = sf.openSession();
        Query query = hsession.createQuery("from SearchProductVO");

        ArrayList list = (ArrayList) query.list();
        request.setAttribute("productList", list);

        return mapping.findForward("display");
    }
}

```

RetriveProductAction.java

we shouldn't use these two approaches. This is because we are writing business logic and DataAccessLogic in Action class execute() method.

note: we should write the hibernate code only in DAO layer

Develop the following form based application to search for the records and display output to the client.we must use MVC2,DAO,DTO,VO,SingleTon designpatterns and Hibernate Template designpatterns.

searchRecords

Pid:

we have

follow the following steps:

step1 and step2 are same as above searchRecords example

Step3: - Develop a DAO class which is responsible to represent the records in the form Objects and add these to ArrayList object. and return arrayList to model.
The following is an example of writing hibernate code in DAO class

```
public class ProductDAO {  
    public ArrayList searchRecords(SearchProductDTO spDTO) {  
        ArrayList list = new ArrayList();  
  
        // Using Hibernate to Search the records in Product table  
        String hqlQuery = "from SearchProductVO";  
        if (spDTO.getPid() != null && !spDTO.getPid().equals("")) {  
            hqlQuery += " where pid='" + spDTO.getPid() + "'";  
        }  
        SessionFactory sf =  
            new Configuration().configure().buildSessionFactory();  
        Session hsession = sf.openSession();  
        Query query = hsession.createQuery(hqlQuery);  
        list = (ArrayList) query.list();  
        return list;  
    }  
}
```

ProductDAO

But here we are calling configure() and buildSessionFactory() every time.to overcome this problem we using SingleTon design pattern

SingleTon design pattern:

If we want carry out any work only once we can use SingleTon design pattern. This design pattern will make sure that this work will happens only once.for that we will use static variable and static blocks. This is because for these members memory will be allocated only once at time of jvm loading the class into jvm's memory.
we declare the constructor as private because no one is allowed to create an object this class.we can call the methods of this call directly by using the class name.

```

public class HibernateSessionFactorySingleton {
    private static SessionFactory objSessionFactory = null;

    static {
        Configuration objConfiguration = new Configuration();
        objConfiguration.configure();
        objSessionFactory = objConfiguration.buildSessionFactory();
    }

    private HibernateSessionFactorySingleton() {
        System.out.println("hibernate sessionfactory object created ");
    }

    public static SessionFactory getSessionFactory() {
        return objSessionFactory;
    }
    public static Session getSession(){
        return objSessionFactory.openSession();
    }
}

```

HibernateSessionFactorySingleton

Template Design Pattern:

When we take any project we try to perform the operations like insert, update, delete and retrieve records. Instead of writing the code repeatedly we develop a template design pattern. This design pattern will help us in removing the redundant code of hibernate. Because of this reason we call it as **HibernateTemplatedesignpattern.**(eg: instead of writing accno, name, date etc.. in our own style daily we will follow one template. that is easy to write the form)

The following is example of **HibernateTemplate** design pattern

```

public class HibernateTemplate {
    public static boolean save(Object object) {
        boolean isStored = false;
        Session hsession = HibernateSessionFactory.getSession();
        hsession.beginTransaction();
        hsession.save(object);
        hsession.getTransaction().commit();
        HibernateSessionFactory.closeSession();
        return isStored=true;
    }
    public static ArrayList load(String hqlQuery) {
        ArrayList arrayList = new ArrayList();
        Session hsession = HibernateSessionFactory.getSession();
        Query query = hsession.createQuery(hqlQuery);
        arrayList = (ArrayList) query.list();
        HibernateSessionFactory.closeSession();
        return arrayList;
    }
    public static void delete(Object object) {
        Session hsession = HibernateSessionFactory.getSession();
        hsession.beginTransaction();
        hsession.delete(object);
        hsession.getTransaction().commit();
        HibernateSessionFactory.closeSession();
    }
}

```

HibernateTemplate

Step3: - Develop a DAO class which is responsible to represent the records in the form Objects and add these to ArrayList object. and return arrayList to model.

```

public class ProductDAO {
    public ArrayList seachRecords(SearchProductDTO spDTO) {
        ArrayList list = new ArrayList();

        String hqlQuery = "from SearchProductVO";
        if (spDTO.getPid() != null && !spDTO.getPid().equals("")) {
            hqlQuery += " where pid='" + spDTO.getPid() + "'";
        }

        list = (ArrayList) HibernateTemplate.load(hqlQuery);

        return list;
    }
}

```

ProductDAO

Step4:- Develop the javabean (Model) class. It has to call the Dao methods .

```

public class ProductJB {
    public ArrayList seachRecords(SearchProductDTO spDTO) {
        ProductDAO objProductDAO = new ProductDAO();
        ArrayList list = objProductDAO.seachRecords(spDTO);

        return list;
    }
}

```

ProductJB

Model class responsibility is create the object to javabean and call the searchRecords(). return this arraylist object to controller. If we want to write any other business logic we will write in this class only.

Step5:- Develop the formbean class and configure that in struts configuration file. In this example we are not developing any formbean class just we are configuring the predefined Formbean class as shown below

```
<form-beans>
    <form-bean name="searchProductForm" type="DynaValidatorForm">
        form-property name="pid" type="java.lang.String" />
    </form-bean>
</form-beans>
```

struts-config.xml

Step6:- Develop the Action class and configure that in struts configuration. The action class responsibility is capture the data and stored it DTO object and call the javabean methods and get the arraylist object and store it in request scope.

```
public class SearchProductAction extends Action {
    public ActionForward execute(ActionMapping mapping,
        ActionForm form,HttpServletRequest req,HttpServletResponse res) {
        DynaValidatorForm dynaForm = (DynaValidatorForm) form;
        SearchProductDTO spDTO = new SearchProductDTO();
        spDTO.setPid((String)dynaForm.getString("pid"));
        ProductJB objProductJB = new ProductJB();
        ArrayList list = objProductJB.searchRecords(spDTO);

        request.setAttribute("productList", list);

        return mapping.findForward("display");
    }
}
```

SearchProductAction

Step6:- Develop the Input form(search.jsp)

Step7:- Develop the success form (display.jsp) which is responsible to get the arrayList object from that display the product details to user.

```

<body>
    <c:if test="${!empty productList}">
        <c:forEach var="SearchProductVO" items="${productList}">
            ${SearchProductVO.pid}
            ${SearchProductVO.name}
            ${SearchProductVO.price}<br>
        </c:forEach>
    </c:if>

    <c:if test="${empty productList}">
        No records found..<br/>
    </c:if>
    <a href="SearchProduct.jsp">Search again</a>
</body>

```

display.jsp

Action Chaining:

Executing the multiple actions in a single request is called as Action Chaining. For example we have developed two action classes, one class has to check whether user is logged in or not then insert the record in database server. Another class also has to check the user is logged in or not then delete the record.

```

public class InsertAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm
        form, HttpServletRequest req, HttpServletResponse res) {
        System.out.println("checking the user login");
        System.out.println("Insert record");
        return mapping.findForward("one");
    }
}
```

InsertAction

```

public class DeleteAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm
        form, HttpServletRequest req, HttpServletResponse res) {
        System.out.println("Check the userlogin");
        System.out.println("Deleting the records");
        return mapping.findForward("two");
    }
}
```

In the above action classes we are trying to check whether user is logged in or not. But this code is provided in multiple Action classes. whenever we want change the we have to change it multiple classes. Because of this we may get project maintainance problems. To resolve these problems **we need to make sure that one Action class has to carry out only one task**. By using this we can remove the redundancy code in project.

- We can configure one action class with different path values.

eg: **<action path="insert" type="InsertAction" />**
<action path="ins" type="InsertAction" />

From the above code we have understand we can use either **/insert.do** or **/ins.do** to execute the InsertAction

- An ActionForward can forward the request to a jsp or another action class. If we want to forward the request to another action class we have to use that action class path in path attribute as shown below

```
<forward name="one" path="/one.jsp" />
<forward name="one" path="/ins.do" />
```

path of the action class

The above ActionForward is forwarding the request to a jsp as well as another class also.

one Action class has to carry out only one task. To get the above requirement we are writing 3 action classes i.e UserCheckAction , InsertAction, DeleteAction
UsercheckAction has to check whether user logged in or not. InsertAction has to insert the record and DeleteAction has to delete the records

```
public class UserCheckAction extends Action {
    public ActionForward execute(...) {
        System.out.println("Checking the Session");
        return mapping.findForward("userCheck");
    }
}
```

eg:

UserCheckAction

```
public class InsertAction extends Action {
    public ActionForward execute(...) {
        System.out.println("Insert the record");
        return mapping.findForward("one");
    }
}
```

InsertAction

```
public class ActionTwo extends Action {
    public ActionForward execute(...) {
        System.out.println("Deleting the records");
        return mapping.findForward("two");
    }
}
```

ActionTwo

To achieve Action chaining we have to configure action classes in struts configuration file.

for example when client send the request to server with **/insert.do** first UsercheckAction has to execute then InsertAction has to execute.

when client send the request the server with **/delete.do** First UserccheckAction has to check whether user is there are not then ActionTwo has to execute to delete the record.

for that we are configuring the action classes as shown below.

```
<action-mapping>
    <action path="/one" type="com.desconit.struts.ActionOne">
        <forward name="two" path="/pages/two.jsp" />
    </action>
    <action path="/two" type="com.desconit.struts.ActionTwo">
        <forward name="two" path="/pages/two.jsp" />
    </action>
    <action path="/insert" type="com.desconit.struts.UserCheckAction">
        <forward name="userCheck" path="/one.do" />
    </action>
    <action path="/delete" type="com.desconit.struts.UserCheckAction">
        <forward name="userCheck" path="/two.do" />
    </action>
</action-mapping>
```

struts-config.xml

What is local forward?

A forward which is specific to an Action class is called as local forward.

```
<action-mapping>
    <action path="/one" type="com.desconit.struts.ActionOne">
        <forward name="two" path="/pages/two.jsp" />
    </action>
</action-mapping>
```

struts-config.xml

This forward object can be access by only ActionOne class in struts configuration file

What is global forward?

A forward object which can be accessible by all the actions are called as global forwards.

```
<global-forwards>
    <forward name="one" path="/pages/insert.jsp" />
</global-forwards>
```

This forward object can be access by any of Action class in struts configuration file

Global Exceptions:

An Exception is a runtime error which disturbs the flow of execution of the program.

Exceptions are two types: 1) Checked Exceptions 2) Unchecked Exceptions

When a program throws checked exceptions we have to handle. We can do that in two ways

1. Maintain try catch in the program itself.
2. Use **throws** keyword.

we have developed two action classes, **ActionOne** and **ActionTwo**. ActionOne class is trying to load a class which is not available in that folder. ActionTwo class is trying to load a file which is not available in that folder.

```
public class ActionOne extends Action {  
    public ActionForward execute(...) throws Exception {  
        Class.forName("Test");  
        return mapping.findForward("one");  
    }  
}
```

```
public class ActionTwo extends Action {  
    public ActionForward execute(...) throws Exception {  
        FileInputStream fis = new FileInputStream("one.txt");  
        return mapping.findForward("one");  
    }  
}
```

ActionTwo

When the client sends the request to above two actions they are throwing an exception to Actionservice() method. Actionservice() checks exception is handled here or not. As exceptions are not handled in that it will throw an exception to server. Server throws an Exception to user directly. But it is not recommended to throw an exception to user instead of that we will display a message to user.

```
<global-exceptions>  
    <exception key="exception.filenotfound" path="/pages/error.jsp"  
               type="java.io.FileNotFoundException" />  
    <exception key="exception.filenotfound" path="/pages/error.jsp"  
               type="java.io.ClassNotFoundException" />  
</global-exceptions>
```

struts-config.xml

when ever an FileNotFoundException is raised errors.jsp will be executed and the output sent to client.

Predefine Actions: (Built-in Actions)

Struts guys has given some predefined Action classes. These Action classes are available in **org.apache.struts.actions** package. The following are some of the predefined actions

1. BaseAction
2. ForwardAction
3. IncludeAction
4. DispatchAction
5. LookupDispatchAction
6. SwitchAction
7. MappingDispatchAction

To work with predefined actions we must use **Struts-extras.jar file**. When we developing action classes based on these classes we must copied the struts-extras.jar into project lib folder.

The problem in struts1.x is using **servlet** (ActionServlet) as a controller. This is because the **url pattern** is configured as ***.do**. So which request ends with .do that requests only handled by the ActionServlet.

When we using jsp's in struts1.x directly jsp are getting executed. This is against to the rules of MVC2 architecture .Jsp's also must access through controller only.

ForwardAction :

By using ForwardAction we can forward the request to jsp through controller. If we doesn't use ForwardAction we have to develop so many empty action classes which forwards the request to one -one jsp. Because of this reason Maintenance of the project becomes difficult. To resolve this problem we use ForwardAction.

The following the configuration of Forwardaction in configuration file.

```
<action path="/home" type="org.apache.struts.actions.ForwardAction"
       parameter="/home.jsp" >
</action>
```

struts-config.xml

If want to execute the home.jsp we have to send the request to server as **/home.do**. Then it ForwardActionexecute() method will be executed. as part of that they will get the parameter attribute value as like this

```
public ActionForward execute(..){
    // Create a RequestDispatcher the correspondingresource
    String path = mapping.getParameter();
    // Let the controller handle the request
    ActionForward retVal = new ActionForward(path);

    return retVal;
}
```

ForwardAction

According theSunMicroSystems documentation it is not recommended to develop a java program with more than 2000 lines of code as well as not recommended to develop a program with very less amount of java code also.

DispatchAction :

DispatchAction make sure that it club multiple action classes into a single action class. If we want to develop multiple formbased applications which does the relevant work we can use DispatchAction.foreg:



If we want to develop the above four forms we can club four action classes into a single Action by using DispatchAction.

Develop an action class with our own methods which takes only four parameters as like execute() method. and these parameter order should be same as in execute().and return type must be ActionForward object only.

```

public class ProductAction extends DispatchAction {
    public ActionForward storeProdut(...){
        DynaValidatorForm dvf = (DynaValidatorForm) form;
        ProductDTO pDTO = new ProductDTO();
        //store the values in dto object from formbean
        pDTO.setPid(Short.parseShort((String) dvf.get("pid")));
        pDTO.setName((String) dynaValidatorForm.get("name"));
        pDTO.setPrice(Double.parseDouble((String) dvf.get("price")));
        //calling the storeProduct() of model class.
        boolean isStored = new ProductJB().storeProduct(pDTO);
        if (isStored) {
            System.out.println("product details are stored.....");
            return mapping.findForward("stored");
        } else {
            ActionErrors aes = new ActionErrors();
            aes.add("store", new ActionMessage("product.store.failed"));
            addErrors(request, objActionErrors);
            System.out.println("Error in Storing the details..");
            return mapping.findForward("failure");
        }
    }
    public ActionForward updateProdut(...){
        System.out.println("product details are stored....");
        return mapping.findForward("update");
    }
    public ActionForward deleteProdut(...){
        System.out.println("product details are stored....");
        return mapping.findForward("delete");
    }
    public ActionForward searchProdut(...){
        DynaValidatorForm dvf = (DynaValidatorForm) form;
        SearchProductDTO sDTO = new SearchProductDTO();
        sDTO.setPid((String)dvf.getString("pid"));
        ProductJB objProductJB = new ProductJB();
        ArrayList<SearchProductVO> list = objProductJB.searchRecords(sDTO);
        request.setAttribute("productList", list);
        return mapping.findForward("search");
    }
}

```

ProductAction.java

step 2:

Configure above action class into struts configuration file by specifying parameter as attribute. The parameter attribute can be any thing.eg:

```
<action path="/store" name="product" type="ProductAction" parameter="mname">
    <forward name="stored" path="/pages/store.jsp" />
    <forward name="failure" path="/pages/StoreProductForm.jsp"/>
</action>

<action path="/update" name="product" type="ProductAction" parameter="mname">
    <forward name="update" path="/pages/store.jsp"></forward>
</action>

<action path="/delete" name="product" type="ProductAction" parameter="mname">
    <forward name="delete" path="/pages/store.jsp"></forward>
</action>

<action path="/search" name="product" type="ProductAction" parameter="mname">
    <forward name="search" path="/pages/DisplayProductRecords.jsp"/>
</action>
```

Struts-config.xml

Step 3:

To execute the specific method of an Action class we use the following url mname as query string.

<http://localhost:8000/pactions/store.do?mname=storeProduct>

here **mname** is the value supplied to that particular action class parameter attribute value. check it in above config file.

To work with DispatchAction end user will not remember the url's . In the jsp's for action attribute value we specify the query String. like below

```
Enter Product details<br>
<html:form action="search.do?mname=searchProdut">
    <font color="red"><html:errors/></font>
    Pid: <html:text property="pid" /><br>
    <html:submit value="Search" />
    <html:reset />
</html:form>
```

SearchProduct.jsp

IncludeAction:

We use IncludeAction as part of migrated projects. The meaning of migration is converting from one technology to another technology.

IncludeAction is much like ForwardAction except that the resulting resource is included in the HTTP response instead of being forwarded to. It is rarely used. Its only significant use is to integrate legacy applications with Struts transparently. Consider a web site that aggregates information from disparate sources – some of which are non-Struts.

The JSP for such a web site consists of <jsp:include>s to include different resources. One of such <jsp:include> that might be as follows:

```
<jsp:include page="/xoom/LegacyServletA" />
```

It is very clear from the value of the *page* attribute that it is a non-Struts resource.

Wouldn't it be better to have a <jsp:include> that pretends as if the resource exists in the current Struts application? It would be ideal if the page include looked as follows:

```
<jsp:include page="/App1/legacyA.do" />
```

The */legacyA.do* cannot be a ForwardAction because it would perform a HTTP Forward to the above resource instead of including the resource in the HTTP response. Since the HTTP Response OutputStream closes (The J2EE jargon for this is the response has been committed) after HTTP Forward, the servlet container cannot process the rest of the JSP and include its response in the OutputStream. Consequently it throws *IllegalStateException* with a message that "*Response is already committed*".

IncludeAction addresses this problem. Instead of forwarding to the specified resource, it includes the resource in the current response. Consequently the output of the LegacyServletA is displayed in the same HTML as that of the Struts application. You have to add the following ActionMapping in the Struts Config file:

```
<action path="/legacyA" parameter="/xoom/LegacyServletA"
type="org.apache.struts.actions.IncludeAction" />
```

The *parameter* attribute indicates the actual resource that has to be included in the response. As mentioned earlier, the use of IncludeAction is limited to including responses from existing Servlet in the current page. This requires the use of <jsp:include> in the page. If your web application is aggregating response from legacy servlet applications, portlets seem to be the way to go.

Majorly Includeaction is used in migration projects. There are some Servlets where we can't modify the code. But we want to use this servlet in struts project. If we use the Servlets directly we are violating the rules of mvc2 architecture. As every request has to go through the controller we take the help of IncludeAction.

```
public class BarCodeGeneratorServlet extends HttpServlet {
    public void doGet(HSR req, HSR resp) {
        System.out.println("Barcode generator doget");
        resp.getWriter().println("Hello Friends hi..");
    }
}
<web-app>
    <servlet-name>Barcode Generator Servlet</servlet-name>
    <servlet-class>BarCodeGeneratorServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Barcode Generator Servlet</servlet-name>
    <url-pattern>/barcodeServlet</url-pattern>
</servlet-mapping>
</web-app>
```

```

<action path="/barcode" type="org.apache.struts.actions.IncludeAction"
parameter="/barcodeServlet">
</action>

```

struts-config.xml

when we want to call the BarcodeServlet from above project we use **/barcode.do**

From the above configuration we have understand that when client has send the request to IncludeAction internally it is dispatching the request to BarCodeGenerator. We have observed that internal code of IncludeAction is using RequestDispatcher to dispatch the request to Servlets and club the output and sent to client.

We can change the struts configuration file name according to project requirement. whenever we change the configuration file name we must change it in web.xml file. If we want to use our own configuration file name we must configure it in web.xml file with **<input-param>** tag with **config** param name.

If we use default struts-configuration file name we no need to configure configuration file name in web.xml

Generally in a project we will have multiple struts configuration files. Most of the projects every module every module we will have a configuration file name as shown below.

ModuleName	Configuration-Filename
default	Struts-config.xml
customer	Customer-config.xml
invoice	Invoice-config.xml

To work with multiple configuration files we must configure all the configuration files in the web.xml file by using comma (,) operator. as shown below.

```

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>ActionServlet</servlet-class>
<init-param>
  <param-name>config</param-name>
  <param-value>/WEB-INF/struts-config.xml,
  /WEB-INF/customer-config.xml,/WEB-INF/invoice-config.xml
  </param-value>
</init-param>
</servlet>

```

web.xml

SwitchAction:

SwitchAction is used to deal with multiple struts configuration files.

Steps:-

1. Develop the multiple Configuration files according to project requirement

ModuleName	Configuration-Filename
default	Struts-config.xml
customer	Customer-config.xml
invoice	Invoice-config.xml

2. Configure all the sturts config files in web.xml file by using module names as shown below.

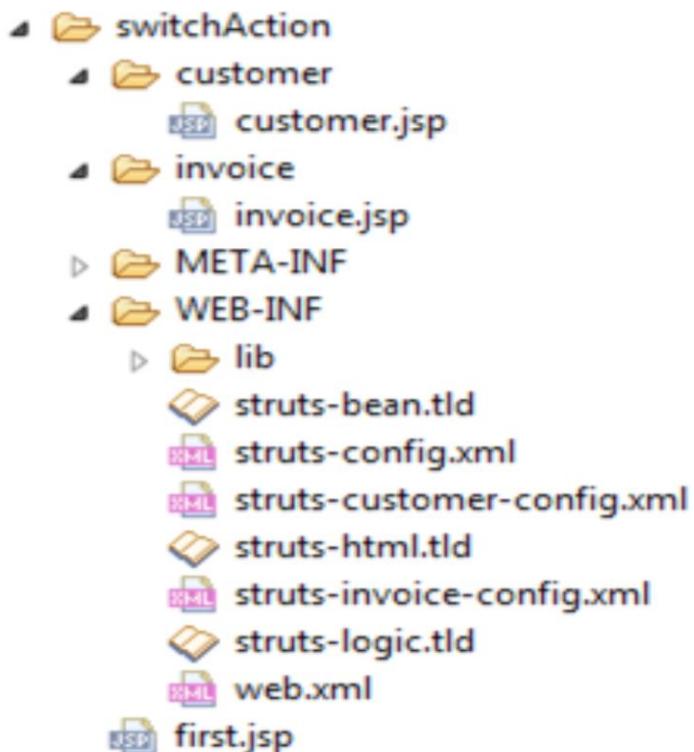
```

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>config/customer</param-name>
    <param-value>/WEB-INF/struts-customer-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>config/invoice</param-name>
    <param-value>/WEB-INF/struts-invoice-config.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

```

web.xml

3. Configure Action classes into appropriate configuration files i.e Configure the module related action classes in that module struts configuration file. Eg: **ActionOne** is related to default module. We have to configure that in default struts configuration file. **CustomerAction** is related to customer module. We have to configure that in **struts-customer-config.xml** configuration file. **InvoiceAction** is related to invoice module. We have to configure that in **struts-invoice-config.xml** configuration file.
4. Create the folders with module names. In this folders we have to place respective module jsp files. as shown below



5. Configure SwitchAction into default struts configuration file.

```

<action-mappings>
  <action path="/default" type="ActionOne" >
    <forward name="first" path="/first.jsp"/>
  </action>
  <action path="/sa" type="org.apache.struts.actions.SwitchAction">
  </action>
</action-mappings>

```

struts-config.xml

```

<action-mappings>
  <action path="/customer" type="CustomerAction">
    <forward name="customer" path="/customer.jsp"/>
  </action>
</action-mappings>

```

struts-customer-config.xml

```

<action-mappings>
  <action path="/invoice" type="InvoiceAction">
    <forward name="invoice" path="/invoice.jsp"/>
  </action>
</action-mappings>

```

struts-invoice-config.xml

6. To execute the Action classes which are belongs to default module we can call them directly like **/default.do**

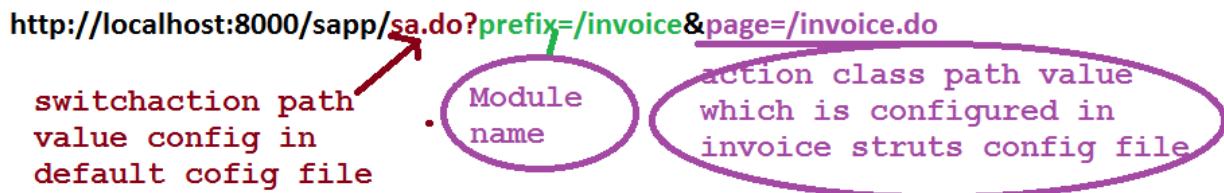
To execute the Action classes which are belongs to other modules like customer and invoice we have to take the help of SwitchAction. To work with switch Action it requires two request parameters. They are

- i) prefix
- ii) page

prefix: we have to supply module name

page: we have to supply the action class path value.

<http://localhost:8000/sapp/sa.do?prefix=/invoice&page=/invoice.do>



Note: Most of the projects uses comma(,) separator to deal with multiple configuration files.

LookupDispatchAction

we use LookupDispatchAction to deal with a form with multiple submit buttons and one action class.

The diagram shows a form layout with three fields: 'Pid', 'Name', and 'Price', each associated with a text input box. Below the fields are two buttons: 'Add' and 'Delete'.

Procedure to work with LookupDispatchAction:

1. Develop an Action class which is sub class of LookupDispatchAction. In this we must provide the implementation to an extra method **getKeyMethodMap()**. This will help us to find which button is mapped with which method.

```
public class ProductAction extends LookupDispatchAction {
    public ActionForward storeProdut(ActionMapping mapping, ActionForm form, HSR,HSR){
        DynaValidatorForm dvf = (DynaValidatorForm) form;
        ProductDTO pDTO = new ProductDTO();
        //store the values in dto object from formbean
        pDTO.setPid(Short.parseShort((String) dvf.get("pid")));
        pDTO.setName((String) dvf.get("name"));
        pDTO.setPrice(Double.parseDouble((String) dvf.get("price")));

        //calling the storeProduct() of model class.
        boolean isStored = new ProductJB().storeProduct(pDTO);
        if (isStored) {
            System.out.println("product details are stored.....");
            return mapping.findForward("stored");
        }
    }
}
```

```

} else {
    ActionErrors aes = new ActionErrors();
    aes.add("store", new ActionMessage("product.store.failed"));
    addErrors(request, objActionErrors);

    System.out.println("Error in Storing the details..");
    return mapping.findForward("failure");
}
}

public ActionForward updateProdut(...){
    System.out.println("product details are stored....");
    return mapping.findForward("update");
}

public ActionForward deleteProdut(...){
    System.out.println("product details are stored....");
    return mapping.findForward("delete");
}

public ActionForward searchProdut(...){
    DynaValidatorForm d vf = (DynaValidatorForm) form;
    SearchProductDTO s DTO = new SearchProductDTO();
    s DTO.setPid((String)d vf.getString("pid"));

    ProductJB p JB = new ProductJB();
    ArrayList list = p JB.searchRecords(s DTO);

    request.setAttribute("productList", list);
    return mapping.findForward("search");
}

protected Map getKeyMethodMap() {
    Map map = new HashMap();
    map.put("button.save", "storeProdut");
    map.put("button.retrieve", "searchProdut");
    return map;
}
}

```

2. Configure the button names in properties files.

button.save = Store
button.retrieve = Search

Message-resources.properties

3. Develop an input form with multiple submit buttons

```

<html:form action="product.do">
    Pid: <html:text property="pid" /><br>
    Name: <html:text property="name" /><br>
    Price: <html:text property="price" /><br>
                parameter value
    <html:submit property="mname" >      of ProductAction
        <bean:message key="button.save"/>
    </html:submit>
    <html:submit property="mname">
        <bean:message key="button.retrieve"/>
    </html:submit>
</html:form>

```

- Configure Action class into struts configuration file by supplying **parameter** attribute.

```

<form-bean name="product" type="DynaValidatorForm">
    <form-property name="pid" type="java.lang.String" />
    <form-property name="name" type="java.lang.String" />
    <form-property name="price" type="java.lang.String" />
</form-bean>
<action path="/product" name="product" type="ProductAction"
parameter="mname">
    <forward name="stored" path="/pages/store.jsp" />
    <forward name="failure" path="/pages/StoreProductForm.jsp"/>
    <forward name="search"
path="/pages/DisplayProductRecords.jsp" />
</action>

```

struts-config.xml

we can develop our own servlet which acts as a controller. If we develop a controller based on HttpServlet we have to provide huge amount of code.

If we want to develop our own controller it's recommended to inherit properties of ActionServlet. we can develop our own servlet which acts as a Controller. compulsory the controller must inherit the properties of ActionSErvlet. As part of service() method we must call ActionServlet service().

```

public class MyServlet extends ActionServlet {
    public void service(HSR,HSR){
        super.service(req,res);
    }
}

```

we would like to develop an application which supports only POST request. if the client sends GET request application should display an error 'GET method is not supports' . The following controller make sure that it block all incoming get request.

```

public class MyServlet extends ActionServlet {
    public void service(HSR,HSR){
        String method = request.getMethod();
        if(method.equals("POST")){
            super.service(req,res);
        } else {
            res.getWriter().println("GET is not supported");
        }
    }
}

```

It's not recommended to develop our own controllers which inherits the properties of ActionServlet. The problem with this approach is improper inheritance between the classes.

RequestProcessor:

As part of ActionServlet service() method they have provided code to create the object to RequestProcessor and call the those methods.

RequestProcessor contains the processing logic that ActionServlet performs the action when ever it get the request. We can customize the RequestProcessor behavior logic by subclassing that class.

The following class override RequestProcessor to check whether the client is sending get request or post request.

```

public class MethodCheckRequestProcessor extends RequestProcessor {
    protected boolean processPreprocess(HSR request, HSR response) {
        if(request.getMethod().equals("POST")){
            return false;
        } else {
            response.sendError(500, "GET method is not supported");
            return true;
        }
    }
}

```

We have to configure the RequestProcessor into struts configuration file.

```

<struts-config>
    <action-mappings>
        <action ..../>
    </action-mappings>
    <controller processorClass="MethodCheckRequestProcessor"/>
</struts-config>

```

struts-config.xml

We say that ActionServlet is the backbone of the struts application but actually it just receives the request and invoke RequestProcessorprocess() method and this RequestProcessor processes all aspects of the request.

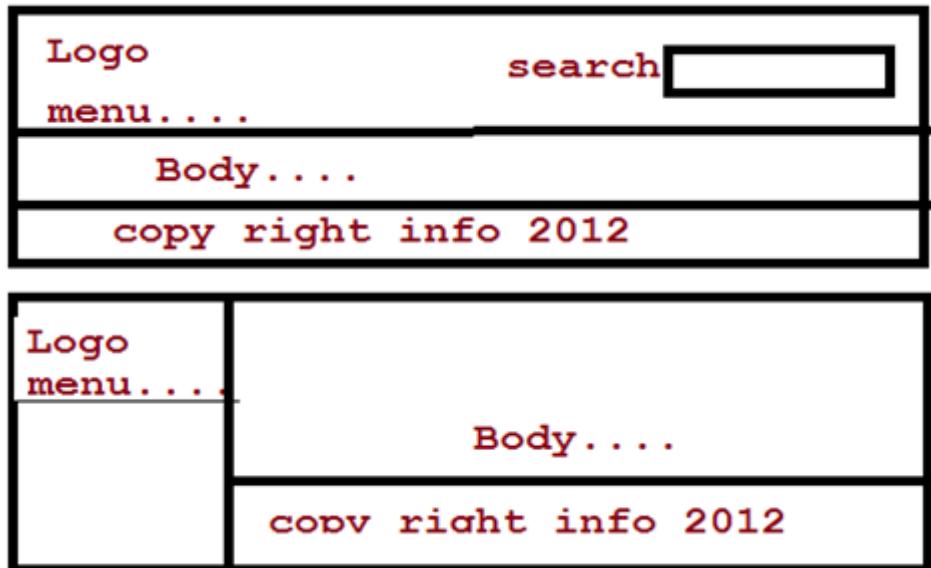
If the above is true , then RequestProcessor should be the heart of struts application or do we consider {ActionServlet + Requestprocessor} as a single unit when we say "ActionServlet is the backbone of the struts application".

1. ActionServlet receives the request.
2. The doPost() or doGet() methods receive a request and invoke the process() method.
3. The process() method gets the current RequestProcessor and invokes the RequestProcessor. process() method
- 4.The RequestProcessor.process() method is where the current request is actually serviced. This method retrieves, from the struts-config.xml file, the <action> element that matches the path submitted on the request. It does this by matching the path passed in the <html:form /> tag's action element to the <action> element with the same path value
5. When the RequestProcessor.process() method has a matching <action>, it looks for a <form-bean> entry that has a name attribute that matches the <action> element's name attribute.
6. When the RequestProcessor.process() method knows the fully qualified name of the FormBean, it creates or retrieves a pooled instance of the ActionForm named by the <form-bean> element's type attribute and populates its data members with the values submitted on the request
7. After the ActionForm's data members are populated, the RequestProcessor.process() method calls the ActionForm.validate() method, which checks the validity of the submitted values.
8. At this point, the RequestProcessor.process() method knows all that it needs to know and it is time to actually service the request. It does this by retrieving the fully qualified name of the Action class from the <action> element's type attribute, creating or retrieving the named class, and calling the Action.execute() method
9. When the Action class returns from its processing, its execute() method returns an ActionForward object that is used to determine the target of this transaction. The RequestProcessor.process() method resumes control, and the request is then forwarded to the determined target.
10. At this point, the ActionServlet instance has completed its processing for this request and is ready to service future requests.

Tiles:

Tiles plugin is used for designing the websites and Layouts. Apache guys has developed tiles plugin and it is integrated with struts. we can use tiles plugin in other frameworks like spring .

The following are sample layout of the project.



Procedure to work with tiles tag library:-

1. Instead of writing the layouts in all the programs we have to write the layout in one program. In this program we have to import the tag library of tiles and define the variables.

```
<%@ taglib uri="http://struts.apache.org/tags-tiles"
   prefix="tiles" %>
<html>
  <body>
    <tiles:insert attribute="header" />
    <tiles:insert attribute="body" />
    <tiles:insert attribute="footer" />
  </body>
<html>
```

layout.jsp

2. Develop header.jsp, footer.jsp, and body.jsp's
3. Develop a jsp which club layout, headers,footer and body..

```
<%@ taglib uri="http://struts.apache.org/tags-tiles"
   prefix="tiles" %>
<tiles:insert page="layout.jsp" flush="true">
  <tiles:put name="header" value="header.jsp" />
  <tiles:put name="body" value="page1.jsp" />
  <tiles:put name="footer" value="footer.jsp" />
</tiles:insert>
```

one.jsp

