CoreJava | JSP | Servlets | JDBC | Struts | Spring | Hibernate

Projects | FAQs | Sample Programs | eBooks | Certification Stuff

Question Banks | Communities | Tutorials | Softwares | Sample Resumes

Interview Tips | Forums | Discussions | Online Test Engines | Jobs

www.JavaEra.com

A Perfect Place For All Java Resources

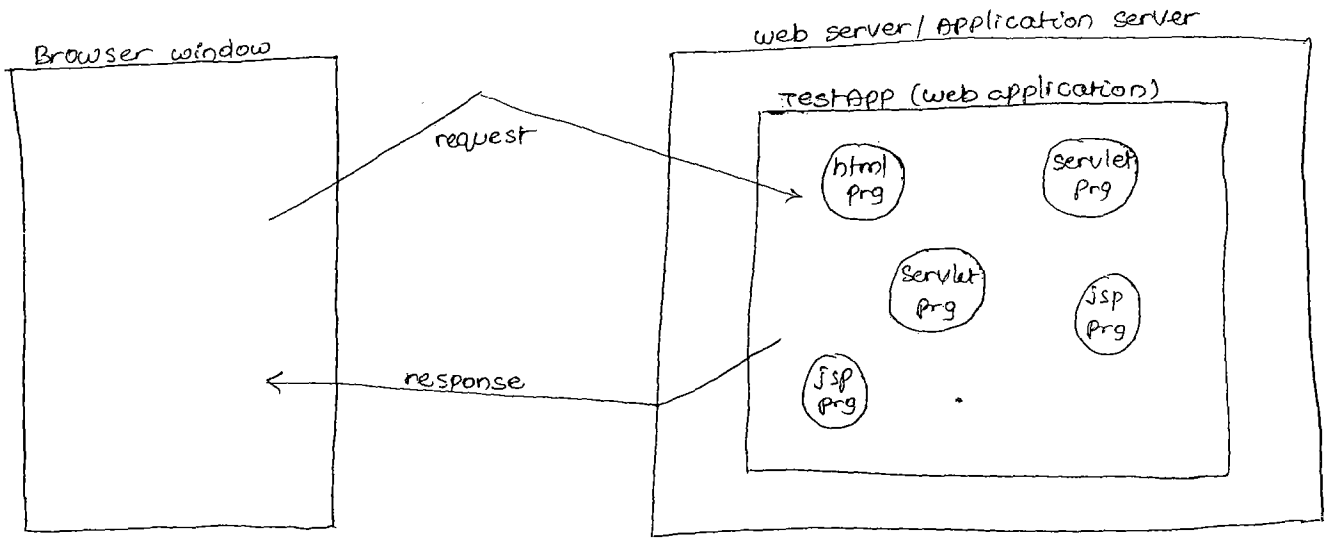
www.JavaEra.com

Nataraz (Sathya)

JSP Notes

JavaEra.com

JSP



* A web application is a collection of web resource programs having capabili to generate web pages. There are two types of web resource programs:
  (1) Server side programs
  (2) client side programs
* server side programs resides and executes from the server.
  Ex: servlet program, JSP program and etc..
* clientside programs resides in web server but they come to browser wir for execution.
  Ex: html program, JavaScript program, VB script program and etc...

NOTE: Decide wheather web resource program is client side or server sid program based on the place where it executes not based on the place where it resides.

* Server side programs generate dynamic web pages.
* client side programs generate static web pages.
* Every web application is a collection of both client side and server side wel resource programs.

client side technologies (useful to develop clientside web resource programs)

html
JavaScript
VB script
Ajax

Server side technologies (used to develop server side web resource programs)

Servlets ——→ from Sun MS (2) } (1)
JSP ——→ from Sun MS (3) }
ASP ——→ from microsoft
asp.net ——→ from microsoft (4)
PHP ——→ from ~~Adobe~~ & Apache foundation (5)
SSJS ——→ from Nestcape
Coldfusion ——→ From Adobe

Why Sun microsystem has given JSP as server side technology when they have already got servlets as server side technology?

(A) while working with asp, PHP technologies we can go for tags based programming. To work with servlets strong java knowledge is required and tags based programming is not possible so no programmer had liked servlets in its initial days. To attract non-java programmers like asp programmers the sun microsystem has given a tag based server side technology called JSP having all the features of servlets.

Drawbacks of servlets Programming

(1) Strong java knowledge is required so not suitable for non-java programmers.

(2) The presentation logic (html code) will be mixed up with java code (Pw.println()) so modification done in one code may effect another code.

(3) writing HTML code in servlet programming is a complex process.

(4) In servlet programming implicit objects are there but we need to write some additional code to get access to those objects.

(5) Modification done in source code of servlet program will be effected only after recompilation of servlet program and reloading of web application in most of the servers.

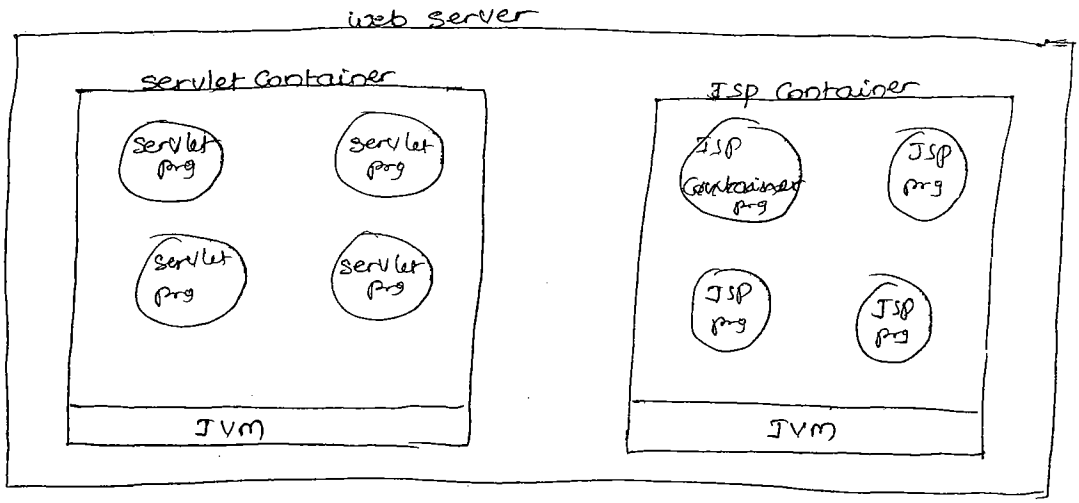
(6) Programmer should explicitly take care of exception handling.

(7) The session object required for session tracking should be created by the programmer manually.

(8) Learning curvies quite high in servlets & servlet programming is not thread safe by default so we need to write some additional code to make our servlet program as thread safe.

Features of JSP

(1) Allows tag based programming so strong java knowledge is not required
(2) suitable for both java and non-java programmers.
(3) use 9 implicit objects and we can use them directly in our JSP program.
(4) Modifications done in JSP program will be recognised by underlaying server automatically without reloading of web application.
(5) takes care of exception handling.
(6) JSP program is thread safe by default.
(7) Allows us to seperate presentation logic from (html code) java code (business logic)
(8) Easy to learn and utilize.
(9) Allows us to work with JSP supplied built-in tags and third party suppli JSP tags and even allows to develop custom JSP tags.
(10) use all the features of servlet.



* JSP, asp, asp.net, PHP programs are there to execute based on page compilation process.
* The process of converting one form of program source code to another for of source code is called as page compilation.
* In JSP page compilation the source code of JSP program will be conver into an equivalent servlet program source code. For this JSP page compiler is required.
* Every web server/application server supplies one JSP page compiler.
* Servlet Container executes Servlet programs and takes care of the life cycle of servlet programs.

* JSP container executes JSP programs by executing the JSP equivalent servlet programs and also takes care of the life cycle of JSP programs.
* Every JSP container is enhancement of servlet container.
* Every web server and application server supplies servlet container and JSP container.
* The servlet container is developed based on servlet api specification. In Tomcat server servlet container name is catalina.jar.
* JSP container will be developed based on servlet and JSP api specification. In Tomcat server JSP container name is Jasper (Tomcat_home\lib\jasper.jar)

The 9 implicit objects of JSP programming

out
config (It is ServletConfig)
application (It is ServletContext obj)
request
response
page
pageContext
exception
session

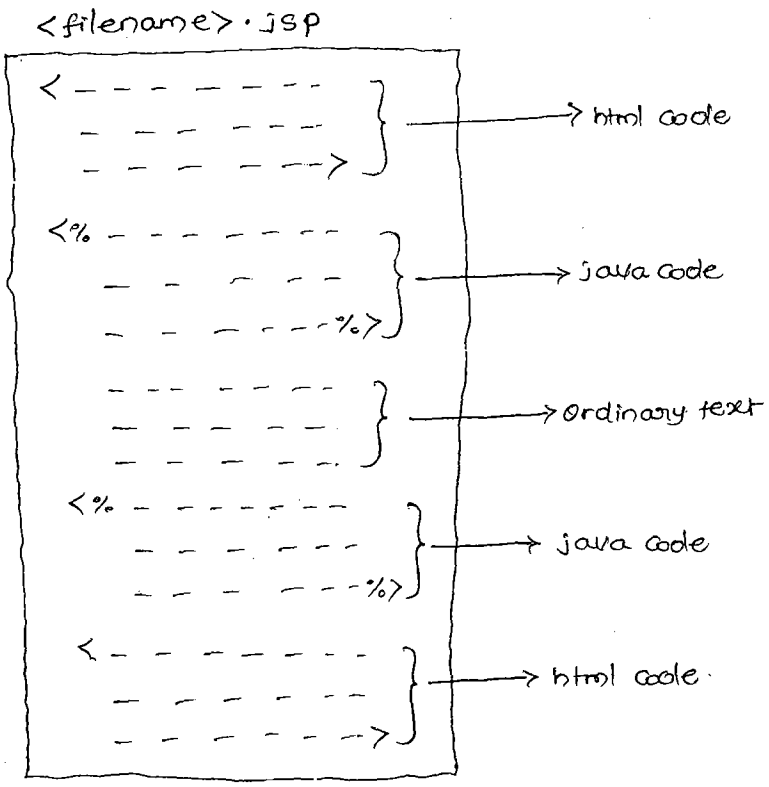
* once the JSP equalent servlet is generated all these objects will be created in that servlet automatically. So we can call them as implicit objects or) built-in objects of JSP.

what is the difference between HTML and JSP ?

HTML	JSP
* client side technology	* server side technology.
* Needs html ~~browser~~ interpreter for execution.	* Needs JSP page compiler and JSP container for execution.
* HTML program generates static web pages.	* JSP programs can generate static & dynamic web pages.
* Given by W3C.	* Given by Sun micro systems.
* Can be used in any kind of web application development. (Java & non-Java web applications).	* should be used only in java based web applications.

* Does not allow to create custom tags. * Allows us to create custom JSP

Structure of JSP program



* <% ---- %> is called as scriptlet.

Template text = html code + ordinary code (text)

Dynamic web page (Generated by JSP program)

Date and time is : nov 25 9:36 am

Fixed content — This is changed time to time

* JSP program can generate dynamic web page. The fixed content of this dynamic web page will be generated through template text of JSP program similarly the dynamic values / content of that dynamic web page will be generated through java code of scriptlet placed in JSP program.

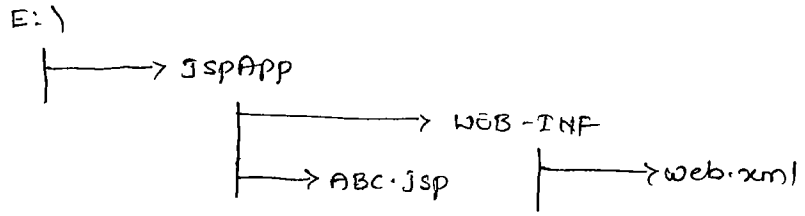
Ex code ABC.JSP

```
<b> <i> Date and Time is : </b><i>          ---> Template text
<% java.util.Date d=new java.util.Date();
   out.println(d.toString()); %>            ---> java code
```

out → implicit object.

Procedure to deploy the Jsp program based web application

step(1): create deployment directory structure by taking the JSP program as web resource program.



ABC.JSP

Same as above code.

web.xml

<web-app/>

NOTE: Configuration of jsp program in web.xml file is optional.

Step(2): Deploy the above web application in tomcat server.

Copy E:\JspApp folder to <tomcat_home>\webapps folder.

Step(3): Start the tomcat server

Step(4): Test the webapplication. Give request to ABC.jsp

open browser window → Type this url

http://localhost:2020/JspApp/ABC.jsp

* while working with only JSP programs based web applications the web.xml file is optional in most of the servers but it is recommended to place web.xml file.
* we can hide the JSP program file name & extension from end users to hide technologies details from end users by providing url pattern to JSP program in web.xml file.
* we can provide url-pattern to JSP program by containing Jsp Program in web.xml file as shown below.

```
<web-app>
    <servlet>
```

```
        <servlet-name> xyz </servlet-name>
        <jsp-file> /ABC.jsp </jsp-file>
    </servlet>
    <servlet-mapping>
        <servlet-name> xyz </servlet-name>
        <url-pattern> /test1 </url-pattern>
    </servlet-mapping>
</web-app>
```

* Using following request urls to generate request to Jsp program

  http://localhost:2020/JspAPP/ABC.jsp

  http://localhost:2020/JspAPP/test1

* WEB-INF is a private directory of java web application so on web server can recognise that directory.

* The web resource programs that are placed ~~in side~~ inside WEB-I (or) its sub folders will be recognised by the web server only when they are configured in "web.xml" file.

  Ex: servlet programs.

* The web resource programs that resides outside the WEB-INF folder can be recognised by the webserver directly so there is no need of configuring those programs in web.xml file.

  Ex: html programs, JSP programs.

* The JSP program can have two types of objects.

  (1) Implicit objects:

  The 9 implicit objects like out, request, response and etc.

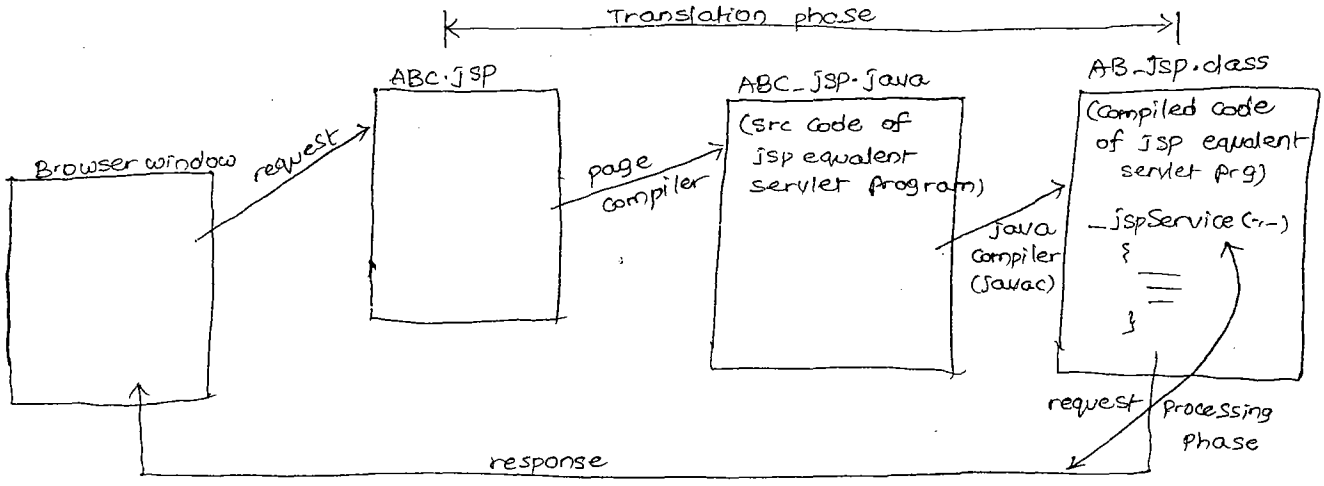
  (2) Explicit objects

  Objects created by Programmer manually.

Ex: <%. java.util.Date d = new java.util.Date(); %>

d ↓ Explicit object

* this, super key words are implicit objects of stand alone java programs.
* request, response, ServletConfig & ServletContext objects are implicit objects of servlet programming and we need to write code to get access to those objects.
* In the execution process of JSP program two phases are there.
  1) Translation phase
  2) Request Processing phase
* In translation page the JSP program will be converted into an equalent servlet program.
* In RequestProcessing phase the compile code of JSP equalent servlet executes and the generated response goes to browser window.



* In Request Processing phase the _jspService(--) of JSP equalent servlet executes and generated response goes to browser window as web page.

The life cycle methods of Servlet Program

init(ServletConfig cg)
service(ServletRequest req, ServletResponse res)
destroy()

The life cycle method of JSP program:

jspInit() / _jspInit() ①, ② are given from tomcat 5 onwards and not there in oth servers as life cycle methods

_jspService(HttpServletRequest req, HttpServletResponse res)

jspDestroy() / _jspDestroy() ③

* _jspInit(), _jspDestroy() methods are introduced as life cycle methods from recent versions of JSP so only latest web server s/w and application server s are supporting them.
* jspInit() / _jspInit() life cycle methods execute when JSP equalent Servlet is instantiated.
* _jspService(-,-) life cycle method executes when JSP program gets request from clients.
* jspDestroy() life cycle method executes when the servletContainer is about to destroy object of JSP equalent servlet program (servlet class).
* In Tomcat server the JSP equalent servlet program of ABC.jsp belonging to JspApp web application comes in <Tomcat_home>\work\Catalina\localhost\Js \org\Apache\JSP folder as ABC_JSP.java, ABC_JSP.class.

  Package name of JSP equivalent servlet class

  ABC_JSP.java → Source code of JSP equalent servlet

  ABC_JSP.class → Compiled code of JSP equalent servlet.
* The file name and class name of JSP equalent servlet program changes server to server.
* The generated JSP equalent servlet is HttpServlet.
* JSP Container internally internally takes the support of servlet Container and servlet life cycle methods call JSP life cycle method and to execute JSP equalent servlet program.
* Every JSP equalent servlet extense from one JSP Container supplied Java class and this java class extent from HttpServlet class and overrides servlet life cycle methods calling JSP life cycle methods internally.
* In Tomcat server JSP equalent servlet class exten from the jasper (JSP container) supplied org.apache.jasper.runtime.HttpJspBase class and this HttpJspBase class extends from HttpServlet class, overrides servlet life cycle methods calling JSP life cycle methods internally.
* init(-) ⟶ internally calls jspInit() / _jspInit()
* service(-,-) ⟶ internally calls _jspService(-,-)
* destroy() ⟶ internally calls _jspDestroy() / jspDestroy()

* refer source code of JSP equalent servlet and .class file once.
* In Tomcat server jasper.jar represents JSP container.
* In weblogic server weblogic.jar represents JSP container, EJB container, servlet container and etc....
* To deploy above given JSPAPP web application in Adv.javaBatchDomain of weblogic 10.3

  copy JSPAPP folder to <weblogic_home>\middleware\user_projects\domains\Adv.javaBatchDomain\autodeploy folder.
* To generate request to JSPABC.jsp of above JSPAPP webapplication use the following request URL.

  http://localhost:7001/JSPAPP/ABC.JSP
* JSP page compiler name in weblogic is jspc and it's generates ABC.JSP related JSP equalent servlet as shown below.

  --abc.java (source code jsp equalent servlet program)
  --abc.class (Compiled code of JSP equalent servlet Program) } For location Search in subfolders of Adv.javaBatchDomai

NOTE: This jspc pageCompiler deletes source code of jsp equalent Servlet Program once compiled code is generated.

* The request given to JSP program participates only in request processing phase when JSP program source code is not modified compare to previous request and .class file of JSP equalent servlet program is not deleted otherwise the request given to JSP program participates in both translation phase, request processing phase.
* JSP container preserves the source code of JSP program internally related to current request until next request comes to that JSP program for comparision purpose. In this process JSP container internally uses ARAXIS, WDIFF kind of comparision tools internally.
* There is no necessaty of configuring JSP equalent servlet program generated by JSP page compiler in web.xml file.
* All implicit objects of JSP will be created automatically in _jspService(-,-) method of JSP equalent servlet program.
* Tomcat server gives servlet-api.jar, jsp-api.jar files representing the Sun micro system supplied servlet, JSP api specifications. The same Tomcat server gives catalina.jar (as servlet container), jasper.jar (as JSP container)

based on that are developed based on servlet, JSP api specifications.

What happens if we modify the source code of JSP equalent servlet program manually ?

(A) The modifications will be effected in the output of JSP program w
(i) JSP equalent servlet program is recompiled.
(ii) When the web application is reloaded.
(iii) If request is given to JSP program without modifying source code.
otherwise the modifications done in source code of JSP equalent servlet program will not be effected.

NOTE: Compile the JSP equalent servlet program generated by page compiler of Tomcat server add the following four jar files to classpath.

jsp-api.jar, servlet-api.jar, jasper.jar, el-api.jar

Available in <tomcat_home>\lib folder.

What is the use of enabling <load-on-startup> on JSP program (or) How to make first request given to JSP program directly participating in request processing phase?

(A) Enable <load-on-startup> on JSP program Configuration done in web.xml file on JSP prog
which makes JSP Container and servlet Container complete translation phase and to perform pre instantiation and pre initialization on JSP equalent servlet program either during server startup or during deployment of web application.

* If you modify source code of JSP program before first request then the entire effect of <load-on-startup> will be wasted.

To enable <load-on-startup> on JSP program

```
<web-app>
    <servlet>
        <servlet-name> x </servlet-name>
        <jsp-file> /ABC.jsp </jsp-file>
        <load-on-startup> 4 </load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name> x </servlet-name>
        <url-pattern> /test1 </url-pattern>
    </servlet-mapping>
</web-app>
```

Can you generate JSP equalent Servlet program outside the web server (or) application server before actually deploying the web application and requesting the web application?

(A) yes we can do by running the JSPpage Compiler tool outside the server at Command prompt. This process is useful to know the syntatical errors of JSP program out side the server.

The weblogic server supplies "weblogic.jspc" tool as JSP page Compiler tool.

Example on weblogic.jspc tool

Step(1): Add <welogic-home>\middleware\wlserver_10.3\server\lib\weblogic.jar file to classpath.

Step(2): Arrange JSP Program

E:\temp
  ⟶ ABC.jsp

Step(3): use weblogic.jspc tool

e:\temp> java weblogic.jspc -compiler javac ABC.jsp

It deletes --abc.java after generating --abc.class (JSP equalent Servlet Prg).

The compiler that has to be used to compile the source file JSP equalent servlet program.

e:\temp> java weblogic.jspc -keepgenerated -compiler javac AB.jsp

It preserves --abc.java even after generating _abc.class file.

* Tomcat server gives org.apache.jasper.JspC class as JSP page Compiler.

Example to work with org.apache.jasper.JspC Tool

(1) Keep your web application ready as shown below.

E:\Temp
  ⟶ JSPAPP
    ⟶ WEB-INF
      ⟶ web.xml
    ⟶ ABC.jsp

(2) Make sure that the following jar files are added to classpath.

servlet-api.jar, jsp-api.jar, jasper.jar, jasper-el.jar (available in <Tomcat_home>\lib folder), tomcat-juli.jar (<Tomcat_home>\bin folder)

(3) Run the tool.

E:\Temp\JSPAPP> java org.apache.jasper.JspC -d . ABC.jsp

org.apache.jasper.JspC: this tool is capable of just generating source code of JSP equalent servlet program

-d .: destination folder to generate JSP equalent servlet program.

JSP tags

* The element name that is surrounded with "<", ">" symbol is called tag.
  <B> ----> Bold tag name
  B ----> Bold element name.
* Java code placed in JSP is called as script code. Javascript code, VB script c place in HTML program is called as script code.
* The code that can't be executed independently and should be embeded wi other technologies based programs for execution is called as script code.

(Tags) Built-in tags/ elements of JSP programming

Scripting elements / tags

① scriptlet (<% ---- %>)
② declaration (<%! ---- %>)
③ Expression (<%= ---- %>)

* scripting tags are there to allowing the programmer to place Java code.

Directive tags/elements

① Page Directive (<%@ page attributes %>)
② Include Directive (<%@ include attributes %>)
③ taglib Directive (<%@ taglib attributes %>)

* Directive tags are given to provide global information to JSP page.

JSP comments

<%---------- %> (for Commenting JSP tags)

Standard Action tags

<jsp : forward>
<jsp : include>
<jsp : useBean>
<jsp : setProperty>
<jsp : getProperty>
<jsp : param>
<jsp : plugin>
<jsp : fallback>
:

* Standard Action tags are given to perform dynamic operations in JSP Programming at request processing phase.

scriptlets

* Every JSP tag can be used with two types of syntaxes.
  (1) Standard syntax
  (2) Xml syntax

Standard syntax of scriptlet

```
<%
   - - -
   - - -   } java code
   - - - -
%>
```

xml syntax of scriptlet

```
<jsp:scriptlet>
   - - - -
   - - - -   } java code
   - - - -
</jsp:scriptlet>
```

* The java code placed in scriptlet goes to _jspService(-,-) method of jsp equalent servlet program as it is.
* Programmers can place business logic or requesting processing logic code in the scriptlets of JSP program.
* Implicit objects of JSP are visible in the scriptlet. In one JSP program we can keep multiple scriptlets.
* Variable declared in JSP comes as local variables of _jspService(-,-) method in jsp equalent servlet so they must be initialized.

Ex: In ABC.jsp (with standard syntax)

```
<% int a=10;
   int b=20;
   int c= a+b;
   out.println("Result is:"+c);
%>
```

In ABC.jsp (with the support of XML standard)

```
<jsp:scriptlet>
    java.util.Date d= new java.util.Date();
    out.println(d.toString());
</jsp:scriptlet>
```

Ex(2): <jsp:scriptlet>

```
int a=10;
int b=20;
if (a<b)
    out.println("a is small");
else
    out.println("b is small");
```

</jsp:scriptlet>

* The above code execution gives Exception because in (a<b) statement "<" is we bee be taken as begining of sub tag having XML (or) JSP meaning. That means it will not be taken as the conditional operator of java code.

Solution(1): (work with standard syntax)

```
<% int a=10;
   int b=20;
   if (a<b)
       out.println("a is small");
   else out.println("b is small");
%>
```

Solution(2): (work with XML syntax along with <![CDAT..]]>)

```
<jsp:scriptlet>
    <![CDATA[ int a=10;
    *1        int b=20;
              if (a<b)
                  out.println("a is small");
              else out.println("b is small");
</jsp:scriptlet>
```

(*1) this "CDATA" makes page compiler to take the body of the tag as text information (java code) and supresses the meaning of xml and JSP for that code.

NOTE: while working with XML Syntax based JSP tags use "<" symbol carefully but there are no problems with ">" symbol.

```
<%. public int sum (int x, int y)
                   { return x+y;
                   } %.>
```

The above java code is invalid as java does not support nested meth definition. But we can place method calls code in scriptlets.

```
<%. String s="hello";
    int leng = s.length();
    out.println("Length is:" +leng);
%>.
```

* while developing jsp based web application, while deploying and executing those web applications there is no necessity of adding any jar files to classpath because there is no need of compiling jsp programs from command prompt.

Declaration

standard syntax

```
<%!
    Decl. of instance variables,
    definitions of user defined methods,
    definitions of jspInit() & jspDestroy() life cycle methods.
%>
```

xml syntax

```
<jsp:declaration>
    ====
</jsp:declaration>
```

* The java code placed in declaration tag comes out side the _jspService() of jsp equalent servlet. So implicit objects are not visible in this declaration tag because all implicit objects are local variables in _jspService() method.

Ex(1): `<%! int a=10; int b=20; %>`

* variables declared in declaration tag will come as the instance variables of the jsp equalent servlet class.

How to differentiate variables in the scriptlet when declaration tag variables and scriptlet tag variable are taken having same name?

(a) use 'this' key word (or) implicit object 'page' to differentiate declaration tag variable from the scriptlet tag variable.

```
<%! int a=10; %>
<% int a=20;
    out.println("local variable (scriptlet) a value is:"+a);
    out.println("instance variable (declaration) a value is:"+this.a); %>
```

Ex(2): keeping user defined java method definitions

```
<%! public int sum(int x, int y)
    {
        return x+y;
    } %>
```

```
<%= out.println("Result is:" + sum(10,20)); %>
```

* while working with xml syntax of declaration tag take care of < sym problem with the support of CDATA.

```
<jsp:declaration>
<![CDATA[
    public int findBig(int a, int b)
      {
        if (a<b)
            return b;
        else
            return a;
      } ]]>
</jsp:declaration>
<%= out.println("Big value is:" + findBig(10,20)); %>
```

* we can also use declaration tag to place JSP init, JspDestroy() life c method definitions in our JSP program. By default the JSP equalent serv class does not provide these JspInit(-), jspDestroy() life cycle methods.

Ex(3):

```
<%! public void jspInit()
    {
      s.o.p("JspInit(): can contain initialization");
    } %>
<%! public void jspDestroy
    {
      s.o.p("Jsp Destroy(): can contain some uninitialization");
    } %>
```

* Jsp Container calls jspInit(-), _jspInit life cycle methods through init(- life cycle method when container instantiate JSP equalent servlet class.
* since JSP equalent servlet automatically supplies _JspInit(-) method # programmer in JSP equalent servlet the programmer should use only jspIni method to place his initialization logic.
* Container calls jspDestroy(), _jspDestroy() methods through destroy() lif cycle methods when Container is about to destroy the object of JSP equalent servlet class.
* since JSP equalent servlet automatically supply _jspDestroy() method the programmer should use jspDestroy() in his JSP program to place un initialization logic.

NOTE: only Tomcat 6.0 onwards _jspInit(), _jspDestroy() methods are introduced in earlier versions of tomcat and in other servers _jspInit(), _jspDestroy() methods are not there.

The overall conclusion is _jspInit(), _jspDestroy() methods are useless methods for programmer.

* we can not define _jspService(--) method in declaration tag of JSP program because jsp equalent servlet automatically gives that metha so our method becomes duplicate method to it and java does not support to have duplicate methods in a java class.

In Jsp Program

```
<%! public void _jspService(HttpServletRequest req, HttpServletResponse res)
    {
    } %>
```

In valid code

Expression tag :

standard syntax

```
<%= java expression %>
```

xml syntax

```
<jsp:expression>
    java expression
</jsp:expression>
```

* This tag evaluates given expression and writes generated result to browser window.
* Code placed in expression tag goes to _jspService(--) of jsp equalent servlet so implicit objects are visible in expression tag.
* Expressions are nothing but arithmetic operations, logical operations and method calls.

```
<% int a=10;
   int b=20;
   out.println(a+b); %>
```

(equals)

```
<% int a=10;
   int b=20; %>
<%= a+b %>
```

Evaluates a+b expression and writes the result to browser window.

* we can also use the expression tag the values of the variables.

```
<% int a = 10;
   int b = 20; %>
A value is: <%= a %>
B value is: <%= b %> Result is:
<%= a+b %>
```

* we can use expression tag to call java methods (both pre defined and user defined).

```
<%! public int sum(int x, int y)
    {
      return x+y;
    } %>
The result is: <%= sum(10,20) %> <br>
<% String s = "hello how r u"; %>
The substring of <%= s %> is : <%= s.substring(0,5) %>
```

* If expression tag is used to instantiate a java class then the default data of the object will be written to browser window as response.

Ex: Date and Time is : `<%= new java.util.Date() %>`

Example on xml based expression tag

```
<jsp:scriptlet>
    int a=10;
    int b=20;
</jsp:scriptlet>
Result is: <jsp:expression> a+b </jsp:expression>
```

* By using xml syntax of expression tag we can not pass expressions that are using "<" symbol, even though CDATA option is used

```
<jsp:scriptlet>
    int a=10;
    int b=20;
</jsp:scriptlet>                                  Invalid code
Result is: <jsp:expression>
        <![CDATA[a<b]]>
    </jsp:expression>
```

* The above problem can be solved with standard syntax of expression t

Ex: <jsp:scriptlet>
int a=10;
int b=20;
</jsp:scriptlet>
Result is: <%= a<b %>

* It is always recommanded to use scripting elements of jsp by following standard syntax.
* The jsp expression tag can evaluate only one expression at a time. It can not be used to evaluate multiple expressions as shown below.

<% int a=10;
int b=20; %>
Results are: <%= a+b, b+a, a-a %>
wrong statement

* we can not write one scripting tag of jsp inside another scripting tag that means we must use these tags as independent tags.

<% int a=10;
int b=20;
<%= a+b %> ——Invalid statement
%>

Procedure to develop JSP based web application by using netbeans IDE.

Step(1): create web Project having name JspApp2

File menu → new project → Java web → web application → next → Project name: JspApp2 → next → server: GlassFish 2.1 → next → Finish

Step(2): Add JSP program to the web pages folder of JspApp2 web application.

Right click on web pages folder → new → JSP → Jsp file name: First → Finish.

First.JSP

```
<%!
    public String getWishMsg(String uname)
    {
        // current date and time
        java.util.Calendar c1 = java.util.Calendar.getInstance();
        //get current hour of the day
        int h = c1.get(java.util.Calendar.HOUR_OF_DAY);
        //generate wish msg
        if (h<=12)
```

```
else if (h<=16)
        return "Good aftn noon" +uname
    else
        return "Good night" +uname.
    }%>
Date and time is : <%= new java.util.Date() %> <br>
<% String uname = "raja";
%> <br> the wish msg is : <%=getwishmsg(uname)%>
```

Step(3): Run the project

Right click on project → Run

Rewriting the above First.jsp with xml syntax

```
<jsp:declaration>
    <![CDATA[
     public String getwishmsg (String uname)
      {
        ==
      } ]]>
</jsp:declaration>
Date and time is: <jsp:expression> new java.util.Date() </jsp:expressi
<jsp:scriptlet> String uname = "raja"; </jsp:scriptlet> <br>
The wish msg is: <jsp:expression> getwishmsg(uname) </jsp:expression>.
```

we can install tomcat server in two ways

(1) through setup file (to installer) (exe)

(2) through zip file

* The Tomcat server that installed through zip file can only configured with netbeans IDE

Procedure to Configure external tomcat with netbeans IDE

Step (1): Install tomcat that is there in zip file mode (apache-tomcat-6.0. binary.zip file) (Extract this file to a folder)

Step (2): Configure the above tomcat server with netbeans IDE.

Tools → servers → Add Server → Tomcat 6.0 → next →

Server location: <Tomcat_home>\ apache-tomcat-6.0.33

The folder that comes after zip file extraction.

catalina Base : <Tomcat_home>\apache-tomcat-6.0.33

User name : admin

Password : admin → Finish → close

step(3): Link Tomcat server to webproject of netbeans IDE

Right click on web project → properties → run → server : Tomcat 6.0 → ok

step(4): Run the web project

Procedure to Configure Adv.JavaBatch Domain server of weblogic 10.3 with netbeans IDE

Tools → servers → Add server → oracle weblogic server → next

Server location: D:\oracle\middleware\wlserver_10.3 → next

Local instances : Admin server (localhost:7001)

username : javaboss

password : javaboss1 → Finish → close

* Link weblogic server with your web project

same as above

* Run the web project.

Implicit objects

request : javax.servlet.HttpServletRequest → request scope

response : javax.servlet.HttpServletResponse → response scope

out : java.servlet.jsp.JspWriter (AC) → Page Scope

session : javax.servlet.http.HttpSession (I) → session scope

page : this (Currently invoking jsp equalent servlet class obj ref) → page scope

pageContext : javax.servlet.jsp.PageContext (C) → page scope

application : Javax.servlet.ServletContext (I) → web application scope

config : javax.servlet.ServletConfig (I) → Page Scope

Exception : java.lang.Throwable (C) → page scope

* All the above given 9 implicit objects of Jsp are the objects of Servlet Container and Jsp Container supplied java classes implementing the above given interfaces or extending from the above given classes and interfaces of Servlet, Jsp apis given by Sun micro systems.

* Jsp programmers never worries and remembers the class names of these implicit objects because they will be changed based on the server we utilize.

Ex. The implicit object out is not the object of javax.servlet.jsp.JspWriter class. It is the object of underlaying JspContainer supplied java class that extends from javax.servlet.jsp.JspWriter class.

In Tomcat server that class name is : org.apache.jasper.runtime.JspWriterImpl

In weblogic server that class name is : weblogic.servlet.jsp.JspWriterImp

* In JSP program we can use 3 types of Comments to comment the Code

(1) Scripting Comments (Java Comments) (// and /* .... */)

given to Comment the java code of scripting elements (<% ... %>, <%! ... %>, <%= ... %>)

(2) Html Comments / Template Text Comments (<!-- ..... -->)

given to Comment the html code and template text code of jsp program.

(3) Hidden Comments / Jsp Comments (<%-- ..... --%>)

given to Comment the JSP tags of JSP programs

* Every Compiler or interpreter generates white spaces for Commented code so we get the feeling the Commented code is not participating in executi

* JSP page compiler recognizes JSP Comments and generates white spaces in the source code of Jsp equalent servlet program.

* Java Comments will be recognised by javac Compiler and keeps white space in .class file.

* HTML Comments will be recognized by HTML interpreter.

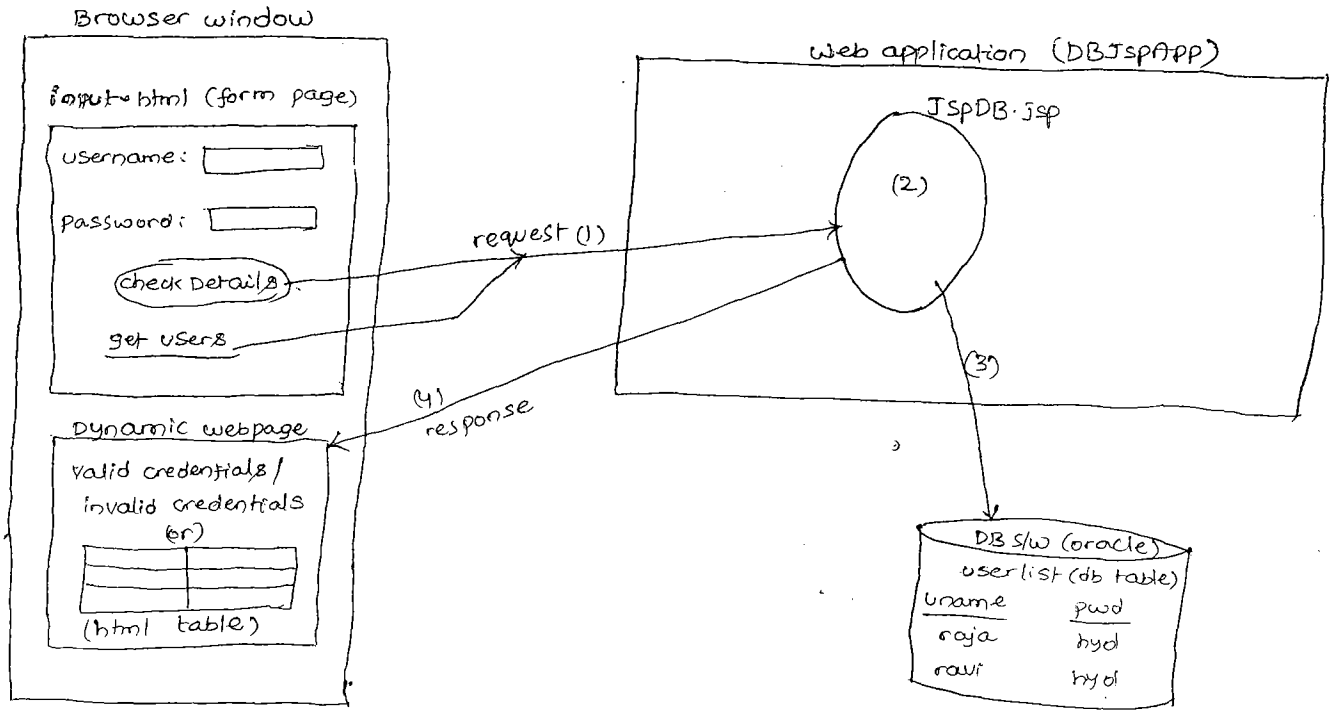
* The JSP program related Jsp Comments are not visible in any phase of JSP programs execution so they are technically called as hidden Co nts.

* we can Comment html code of JSP program with JSP Comments but not recommended. we can Comment jsp tags with html Comments but it is also not recommanded.

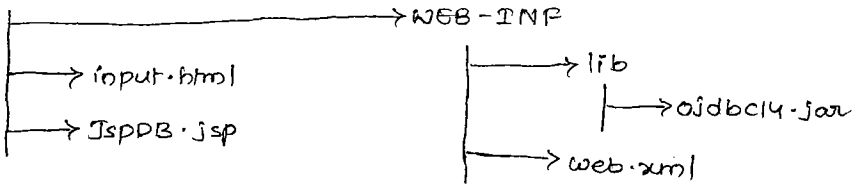
comment type	In src code of Jsp equalent servlet	In Compiled code of jsp equalent servlet	In the html code that goes to browser window	in the output/ response
Jsp Comment <%-- ..... --%>	no	no	no	no
html Comment <!-- ... -->	yes	yes	yes	no
Java scripting Comment (// or /* --- */)	yes	no	no	no

* The HTML program related web page can have form or hyperlink to generate request to JSP or servlet program. This provides GUI ness to end user to Communicate with servlet/ JSP program from browser window
* For HTML to JSP Communication you can use the JSP file name or its URL pattern as action url of form page (as action attribute value of form tag) or as href of anchor tag.
* For JSP to DB Communication place Jdbc code in JSP program.

Example application on html to JSP and JSP to DB s/w Communication



DBJspApp



* The code that comes to _JspService(-,-) method of a JSP equalent ser will be automatically handling the exceptions because that _JspService(- method code will be there surrounded with try and catch blocks.
* the code placed in declaration tag of JSP program must handle the exce explicitly where as the code placed in scriptlet, expression tag automatica handles the exception.

source code of the above diagram based application

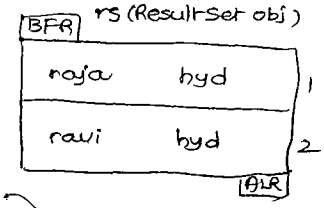
input.html

```
<form action="JspDB.jsp" method="get">
    Login username: <input type="text" name="tname"/><br>
    Login password: <input type="password" name="tpwd"/><br>
    <input type="submit" value="check Details" name="s1"/>
</form><br><br><br>
<a href="JspDB.jsp?s1=link">Get All User Details</a>
```

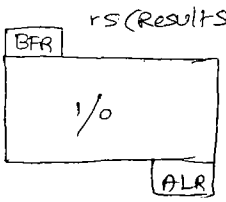
JspDB.jsp

```
<%@ page import="java.sql.*;" %>  //taken to import the java packages.
<%!
    Connection con;
    PreparedStatement ps1, ps2;
    public void JspInit()
    {
      try
        {
          Class.forName("oracle.jdbc.driver.OracleDriver");
          con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:
                                            sathya","scott","tiger");
          ps1 = con.prepareStatement("select count(*) from userlist where
                                      uname=? and pwd=?");
```

```
        PS2 = Con.prepareStatement ("select * from userlist");

      }//try
      catch (Exception e)
       {
        e.printStackTrace();
       }
    }//jspInit()

%>

<% //read form data from form page
     String user = request.getParameter ("tname");
     String pwd = request.getParameter ("tpwd"); //read additional req param value
     String param = request.getParameter ("s1");

     if (param.equals ("link")) //when hyperlink is clicked
     {
       //write jdbc code to get all users details

        ResultSet rs = PS2.executeQuery();
        out.println ("<table>");
        While (rs.next())
         {
           out.println ("<tr>");
           out.println ("<td>" + rs.getString(1) + "</td>");
           out.println ("<td>" + rs.getString(2) + "</td>");
           out.println ("</tr>");
         }//while
        out.println ("</table>");
        rs.close();
     }//if
     else //when submit button is clicked
     {
       //write jdbc code for Authentication.

        //read form data
         String user = request.getParameter ("tname");
         String pass = request.getParameter ("tpwd");
        //set form data as param values of query
         ps1.setString (1, user);
         ps1.setString (2, pass);
        //execute the query
         ResultSet rs = ps1.executeQuery();
```



logic to display ResultSet obj data as html content



```
                int cnt=0;
                if (rs.next())
                    cnt = rs.getInt(1);
                if (cnt==0)
                  out.println("<b><i><font color=red>Invalid credentials </fon
                                                                    </i></b>");
                else
                  out.println("<b><i><font color=red>valid credentials </font></i
              }//else
          %>
          <%! public void jspDestroy()
                {
                  try
                    {
                      ps1.close();
                      ps2.close();
                      con.close();
                    }
                  catch(Exception e)
                    {
                      e.printStackTrace();
                    }
                }//jspDestroy()
```

web-xml

```
<web-app/>
```

* Keep ojdbc14.jar file in WEB-INF\lib folder.
* Deploy the DBJspApp in Tomcat server.
* use the following request URL to test the application.

http://localhost:2020/DBJspApp/input.html

userlist (DB table)

uname	pwd
raja	hyd
ravi	hyd

select * from userlist where uname='raja' and pwd='hyd'

count(*)
1

——> valid credentials

select * from userlist where uname='rajal' and pwd='hyd'

count(*)
0

——> Invalid credentials.

Page Directive Tag

* This tag is given to provide global information to JSP program like buffer size, package import statements, content type and etc...

standard syntax

<% @ page attributes %>

xml syntax

<jsp:directive.page attributes />

attributes

language = "java" default → java // java is the default and only one language that you can pass here as a value.

* The above attribute specifies the language that we need to use to write code in scripting tags. (only java is allow).

import = "java.util.*, java.net.*, ..." //default : java.lang.*

* useful to import the java packages that are required for java application jsp program.

extends = "class name"

* Not recommended to use, no default value.
* Generally the jsp equalent servlet class extends from one or other jsp container supplied predefined java class. If you want to see that jsp equalent servlet class extending from your own or your choice java class then use this extends attribute. Since programmer can not develop that full fledged java class should act as super class of jsp equalent servlet class so this extends attribute is not recomended to use.

contentType = "text/html" //resp content type, // no default value

* This is useful to specify content type of the response that should be generated by JSP program.

Ex: <% @ page language = "java" import = "java.net.*, java.sql.*" contentType = "text/html" %>

buffer = "18kb" or more //default → 8kb

* specifies the size of the buffer at server side.

* Even though the code of JSP program is executing part by part, because of underlaying CPU algorithms (like round robin algorithm) the server presents output of the executed statements in buffer of the JSP program upto until last statement of JSP program gets executed.

Ex: <%@ page buffer = "10kb" %>
 hello
↳ The buffer size of JSP program at server side

Ex: <%@ page buffer = "none" %>
↳ makes the JSP program running without buffer.

NOTE: Every servlet program also contains one default server side buffer But there is no default size for this buffer.

* The server side buffers servlet/JSP program is no way related with client side (browser window) buffer.

* Buffer is a temporary memory that can hold data for temporary period

autoFlush = "true" (false) //default → true

* It cleans the buffer of JSP program and sends to browser window as web page

Ex: <%@ page buffer = "8kb" autoFlush = "true" %>

session = "true" (false) //default → true

* Enables JSP equalent servlet create or not to create implicit object session

Ex(1): <%@ page session = "false" %>
↳ implicit object session will not be created.

Ex(2): <%@ page session = "true" %>
↳ Implicit object session will be created.

* use when we don't enable session tracking we don't need session object in JSP program. To disable that session object creation use above statement

errorPage = "url" //no default
isErrorPage = "false" (true) //default → false
} Required to configure error pages for JSP program to handle the exceptions raised in JSP program.

isThreadSafe = "true" (false) //default → true

* useful to enable or disable thread safty on JSP equalent servlet

Ex: <%@ page isThreadSafe = "false" %>
make JSP equalent servlet as Thread Safe servlet by implementing javax.servlet.SingleThreadModel (i)

info = " String

* The short desc about JSP page - no default value

Ex: <%@ page info="basic program" %>

isELIgnored = "true" (false) //default → false

* To perform arithmetic and logical operations in JSP program without using java code, we can use EL (expression Language).
* The above attribute can make page compiler to ignore or recognise that EL.

Rules to remember while working with page directive tag (<%@ page. %>)

(1) Page directive tag name and attribute names are case sensitive.

(2) Unknown attributes will not be recognised.

(3) Except import attribute no other attribute can have multiple values seperated with "," symbol

<%@ page import="java.net.*, java.awt.*" buffer="8kb, 10kb" %>

It is allowed

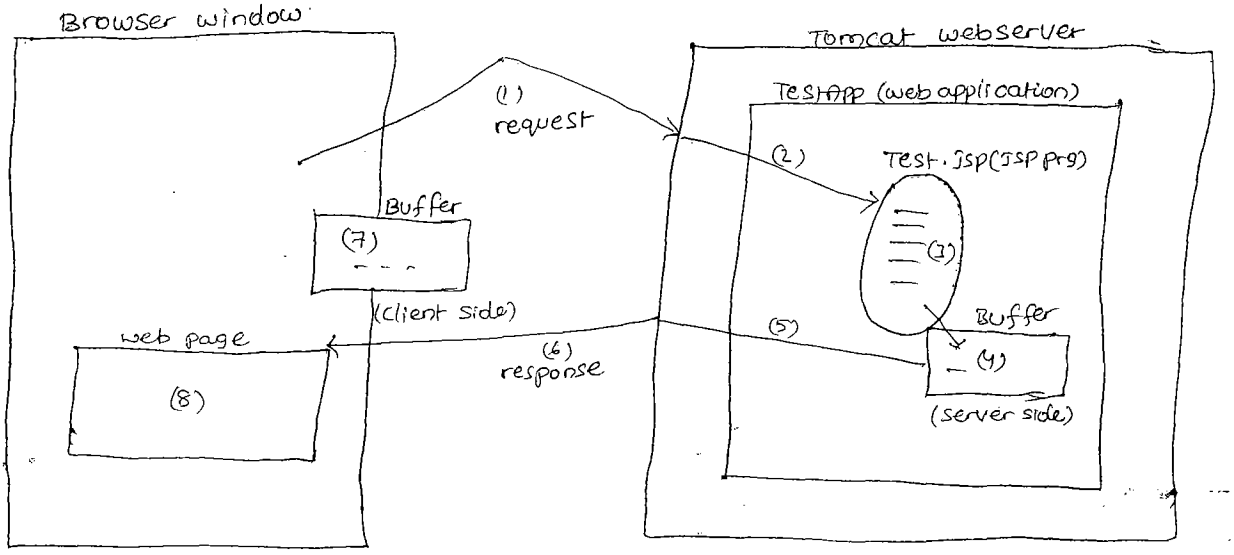
It is not allowed.

(4) Except import attribute no other attribute can not occur for multiple times, with different values either in the same page directive tag or in the multiple page directive tags of same JSP program.

<%@ page import="java.net.*" import="java.net.*" buffer="8kb" buffer="10kb" %>

allowed

In valid



* The JSP program buffer size is taken as 1kb but that JSP program is generating more than 1KB output then what happens?

(A) The JSP program dynamically expandes its size as needed.

NOTE: Even though buffer = none is taken the JSP program is capable of expanding Buffer size as needed at runtime so we can enable autoflush = "true" eventhough buffer is none.

Ex: <%@ page buffer = "none" autoFlush = "true" %>

What is the difference between printWriter() and JspWriter()?

* The process of story result/output in a buffer before devlivering to the orignal destination is technically called as buffering operation.
* PrintWriter can not perform buffering while sending its output to browse window where as JspWriter can perform buffering.
* when JSP program is taken with out buffer then JspWriter internally use PrintWriter to write the output to browser window.
* JspWriter is the type of the implicit object called out in JSP Program
* PrintWriter is useful in servlet Program.
* PrintWriter belongs to java.io package and JspWriter belongs to javax.servlet.jsp package

—•—

* Servlet api is given as

  javax.servlet package, javax.servlet.http package

  Latest version servlet api is : 3.0 (supports Annotations based programmin
  running version of servlet api is : 2.4/2.5

* JSP api is given as

  javax.servlet.jsp package
  javax.servlet.jsp.el package
  javax.servlet.jsp.tagext package

  Latest version of JSP api is 2.1
  running version of JSP api is 2.0

* JSP equalent servlet automatically handles the exceptions but displays the exceptions related error messages quite ugly on browser window when exception is raised. Since these are technical messages the non-technical end users can not understand them. To overcome this problem configure error pages for JSP program.

* Error page is a JSP or HTML program that are capable of executing only when exception is raised in other JSP programs. These are also useful to display the exception related messages non technical messages guiding the end users.
* Always develop webapplications by keeping non-technical end users in mind. For that error pages configuration is important.

Error page Configuration in JSP Programs

1. Local Configuration

→ This error page cfg is specific to one JSP prg.

→ use errorPage, isErrorPage attributes of <%@page...%>

2. Global Configuration

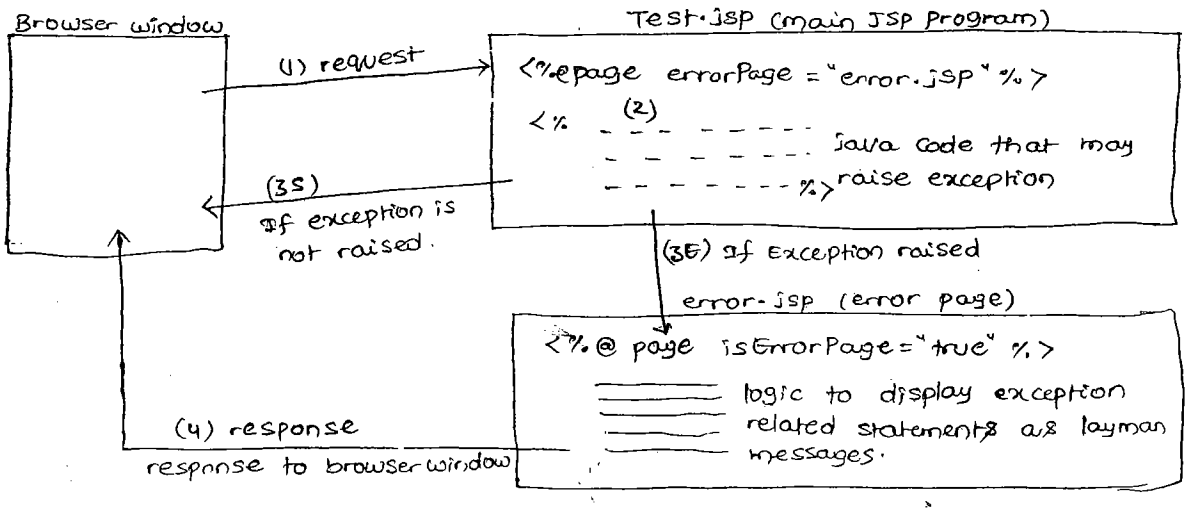
→ this error page cfg is common for all the jsp prgs of a web application

→ use <error-page> of web.xml file

In servlet program we take support of rd.forward(-,-) for Error servlet/ Error page Configuration.

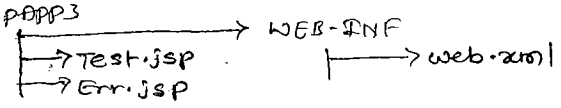
* The ErrorPage Configuration done in web.xml file for JSP programs (global Configuration) will not work for servlet programs of web application.

Local Error page Configuration



* isErrorPage = "true" attribute of page directive tag can make a JSP program of web application as error page.
* The Implicit object Exception is visible only in that JSP program that acts as error page and we can use this object to know the details of exceptions that are raised in main JSP programs.

Example application



Test.jsp

```
<%@ page errorPage="Err.jsp" %>
<b> from Test.jsp </b> <br>
<% int x=Integer.ParseInt("a123");
    out.println("x value is:"+x); %>
```

Err.jsp

```
<%@ page isErrorPage="true" %>
<b> from err.jsp </b><br>
<font color=red>Internel problem </font> <br>
The exception that is raised is: <%= exception.toString() %>
```

Implicit object

web.xml

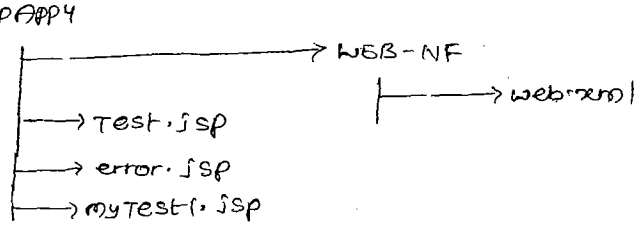
```
<web-app/>
```

Request URL to test the application

http://localhost:2020/Struts App3/Test.jsp

* The above exception handling configuration also allows to take .html file as error page but we can not choose to work with implicit object except in that error page.

Example application on global error page configuration



Test.jsp

```
<b> from Test.jsp </b><br>
<% int x = Integer.paseInt("a123");
    out.println("x value is:"+x); %>
```

MyTest1.jsp

```
<b> from myTest.jsp </b><br>
<% Class.forName("java.lang.System1");
 %>
```

Error.jsp

* Same as the above Err.jsp

web.xml :

```
<web-app>
   <error-page>
      <exception-type> java.lang.Exception </exception-type>
      <location> /err.jsp (or .html) </location>
   </error-page>
</web-app>
```

NOTE: Because of the above configuration done in web.xml file for any exception raised in any Jsp program of web application the control goes to error.jsp page.

* Global error page configuration also allows to take .html program as error page.
* If both local error page, global error page configurations are done in same page to handle same exception pointing to two different error pages can you tell me which settings will be effected ?

(A) The configurations done in local error page configuration will be effected.

Directive include tag

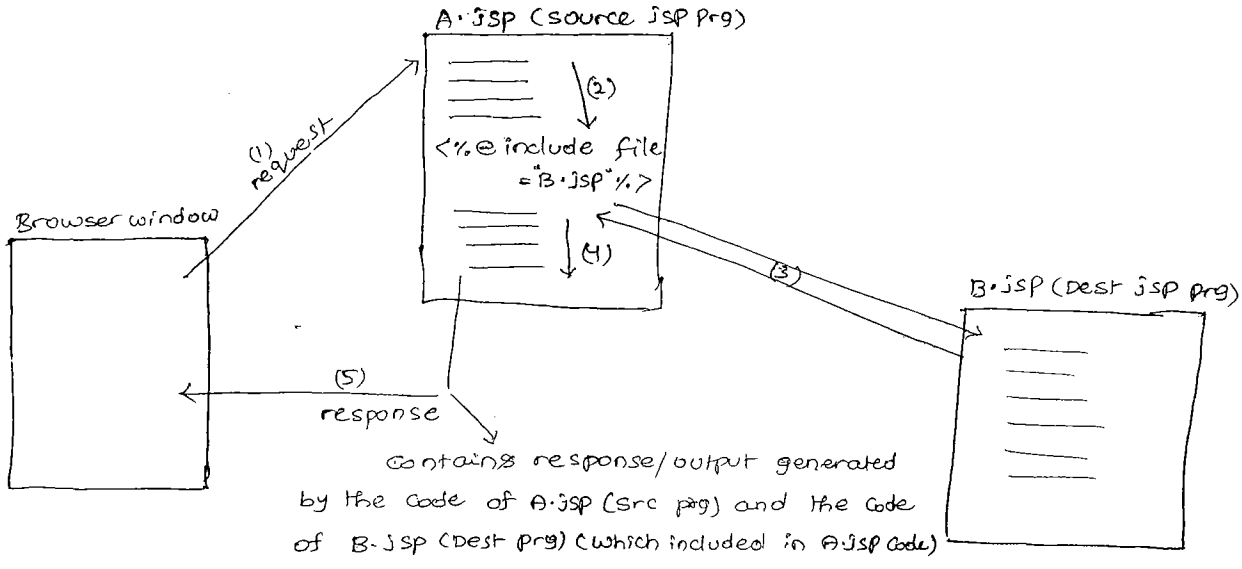
standard syntax :

```
<%@ include file="dest url" %>
```

xml syntax

```
<jsp:directive.include file="dest url" />
```

* This tag is given to include the code or content of destination web resource program to the source code of Jsp equalent servlet belonging to the source Jsp program.

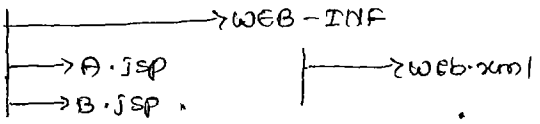


here B.jsp will not be executed seperately but its code will be included to the code of A.jsp and executes along with A.jsp. (Refer step (3) of diagram).

* Directive Include (<%@ include ... %>) does not perform output inclusion bu it performs code/content inclusion.

Example:

JspApp5



A.jsp

```
<b> from A.jsp </b> <br>
<%@ include file="B.jsp" %>
<b> from end of A.jsp </b>
```

B.jsp

```
<b> from B.jsp </b><br>
<%= new java.util.Date() %>
```

web.xml

```
<web-app/>
```

Request URL to test the application

http://localhost:2020/JspApp5/A.jsp

* when the above A.jsp is requested

(a) JSP page compiler generates JSP equalent servlet for A.jsp

(b) Because of <%@ include file="B.jsp" %> the code of B.jsp will be inclu to A.jsp related jsp equalent servlet program.

(c) The java compiler compiles JSP equalent servlet source code of A.jsp and executes that code then sends response to browser window.

* In the above execution B.jsp program will not be executed seperately so its JSP equalent servlet will not be generated seperately.

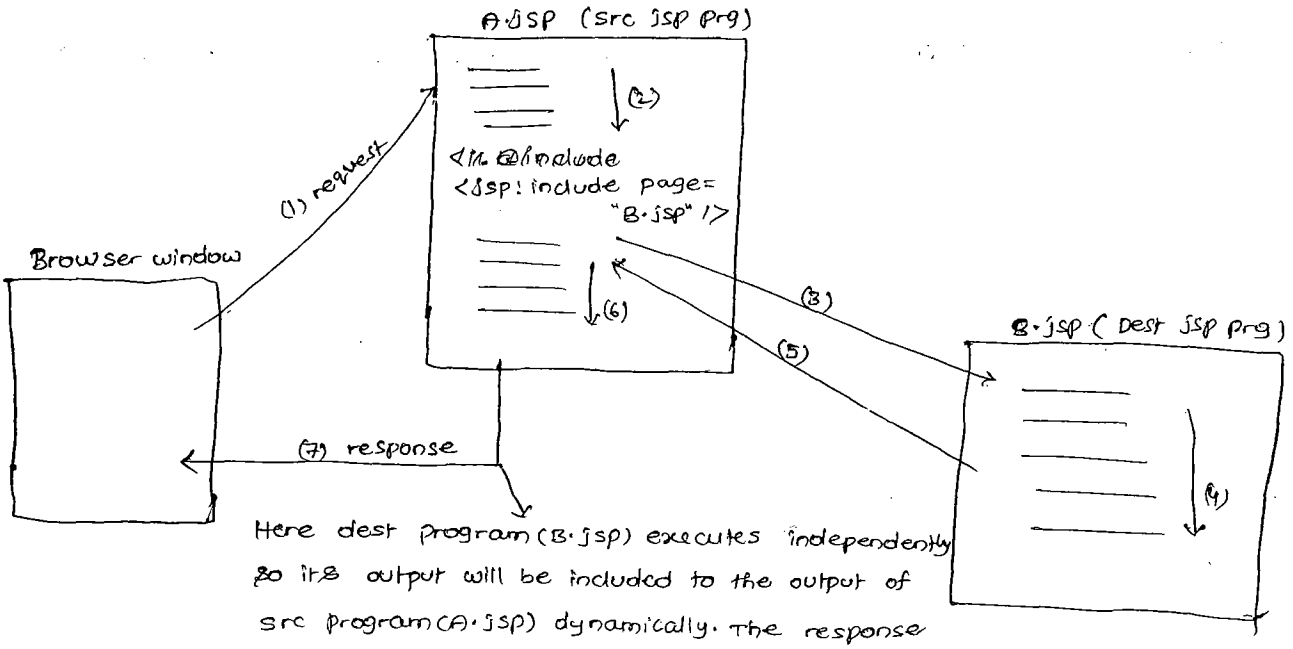
Action Include (<jsp:include>)

Syntax

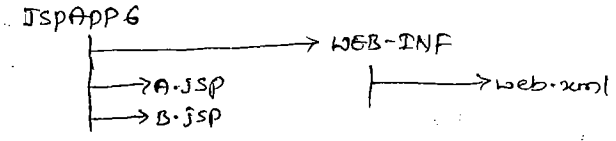
```
<jsp:include page="dest url"/>
```

NOTE: All standard Action tags of JSP like <jsp:include> are having only xml syntax and there is no standard syntax for them.

* <jsp:include> is given to include the output of destination web resource program to the output of to the source JSP program.



Example application



A.jsp

```
<b> from A.jsp </b><br>
<jsp:include page="B.jsp"/>
<b> from end of A.jsp </b>
```

B.jsp

same as B.jsp in JspApp5

web.xml

```
<web-app/>
```

URL to test

http://localhost:2020/JspApp6/A.jsp

* <jsp:include> performs output inclusion but not the code inclusion.
* <jsp:include> is tag based alternate for rd.include(--) method of servlet programming.
* when the above A.jsp is requested two seperate jsp equalent servlets will be generated for both A.jsp, B.jsp and the jsp equalent servlet of A.jsp internally uses some rd.include(--) method to include the output of B.jsp.

* The action tags of JSP can be used only with xml syntax Compare to other tags of JSP (other tags can be having both xml and standa syntax).

* Standard Action tag of JSP are given to perform more dynamic operation at runtime/request processing phase operation.

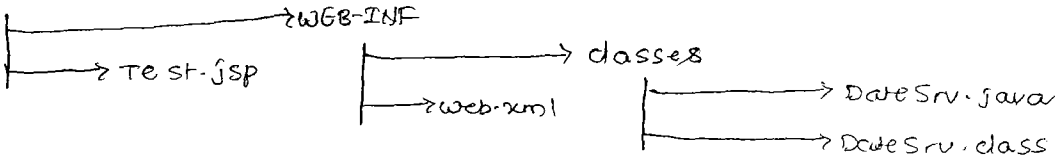
What is the difference between "include Directive" and "action include tag

Include Directive <%@ include ....%>	Action Include <jsp:include>
* given to perform code inclusion.	* given to perform output inclusion
* performs code inclusion at translation phase so this work is called as compile time binding or static binding.	* performs output inclusion at run time/request processing phase & this work is called as dynamic, runtime binding.
* If destination program is JSP the sepeate JSP equalent servlet will not be generated for that program.	* will be generated.
* can not include code and output of servlet program.	* can include (includes the output
* suitable to take static web resource programs as destination programs. (like html programs)	* Suitable to take dynamic web resource programs as destination pro (like servlet program, JSP program)
* Does not use rd.include(-,-) internally.	* uses rd.include(-,-) internally.
<%@ include file="B.jsp" %>	<jsp:include page="B.jsp" />

Example application on JSP program ~~Utilities~~ including the output of servle

Program

JspApp7



Request URL

http://localhost:2026/JspApp7/Test.jsp.

DateSrv.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class DateSrv extends HttpServlet
 {
   public void service(HttpServletRequest req, HttpServletResponse res) throws
                                                    ServletException, IOException
    {
      //general settings
       PrintWriter pw=res.getWriter();
       res.setContentType("text/html");
       //write b.logic
       java.util.Date d=new java.util.Date();
       pw.println(d.toString());
    }//service(-,-)
 }//class
```

web.xml

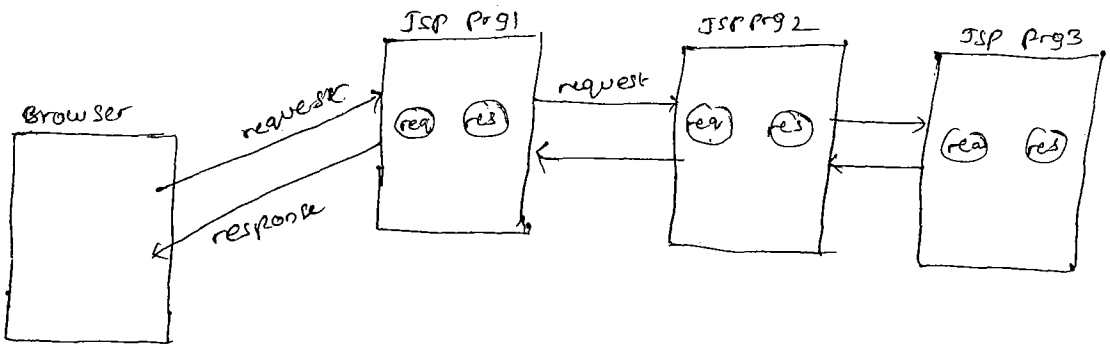
Configure DateSrv Servlet program with test1 url pattern.

Test.jsp

```
<b> from begining of Test.jsp </b> <br>
Date and Time is <jsp:include page="test1" /> <br>
<br>
<i> end of Test.jsp </i>
```

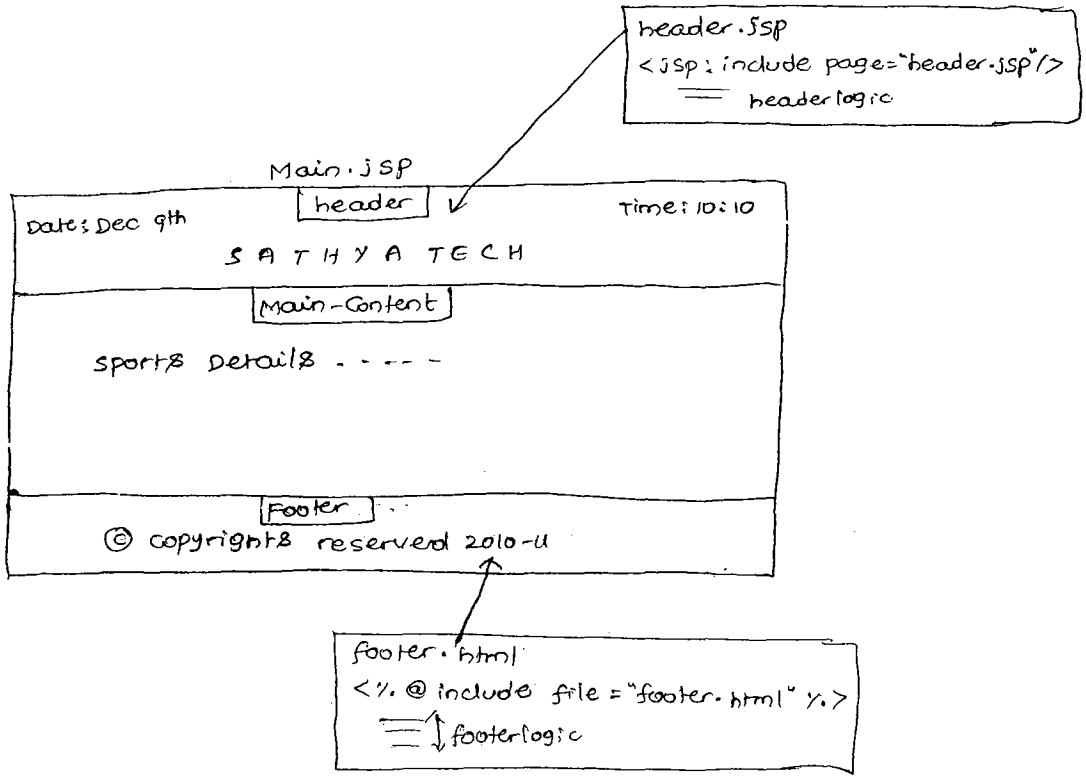
we can not replace this tag with include directive tag

* while working with directive include and action include don't commit the response in destination web resource program by using out.close() or pw.close() methods because it does not allow to add further output of source JSP program to the response.
* Taking a request from a browser window and process the request by using multiple JSP programs as chaining. For this we can use <jsp:include> tag or <jsp:forward> tag.
  * All the JSP programs which participate in a JSP chaining will use same request and response objects.



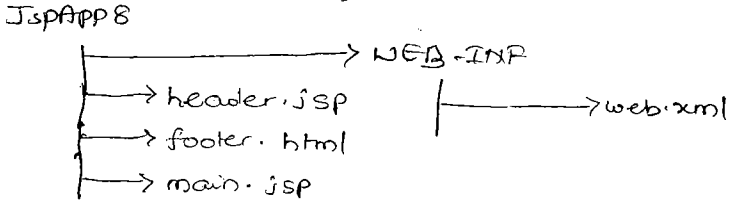
Java program participating in JSP chaining, all the steps JSP Prog of jsp chaining will use same request and response objects.

* If the content to include is static content use directive include tag. If the content to include is dynamic content use action include tag.



* In one JSP program we can see the utilization of both directive include action include tags for multiple times.

Example application based on the above diagram



header.jsp

```
<%@ page import="java.util.*" %>
<% Calendar cl = Calendar.getInstance();
   int date = cl.get(Calendar.DAY_OF_MONTH);
   int month = cl.get(Calendar.MONTH); %>
<%= date %>/<%=month %>      <%=cl.getTime() %>
<br><br>
<marquee> <font color=red size=5> S A T H Y A  T E C H </font> </marquee>
<br>
<hr>
```

footer.html

```
<hr>
<br><br> <center>
&copy; copy rights reserved 2010-11 </center>
```

main.jsp

```
<jsp:include page="header.jsp"/>
<br><br>
Sports news of the day <%=new java.util.Date() %>;
   <font size=2> Sehwag has scored 219 runs in an innings of one day
      cricket match beating sachin's record of 200. </font> <br><br><br>
<%@ include file="footer.html" %>
```

web.xml

```
<web-app>
```

Request URL to test the application

http://localhost:2020/~~strutsApp~~JspApp8/main.jsp

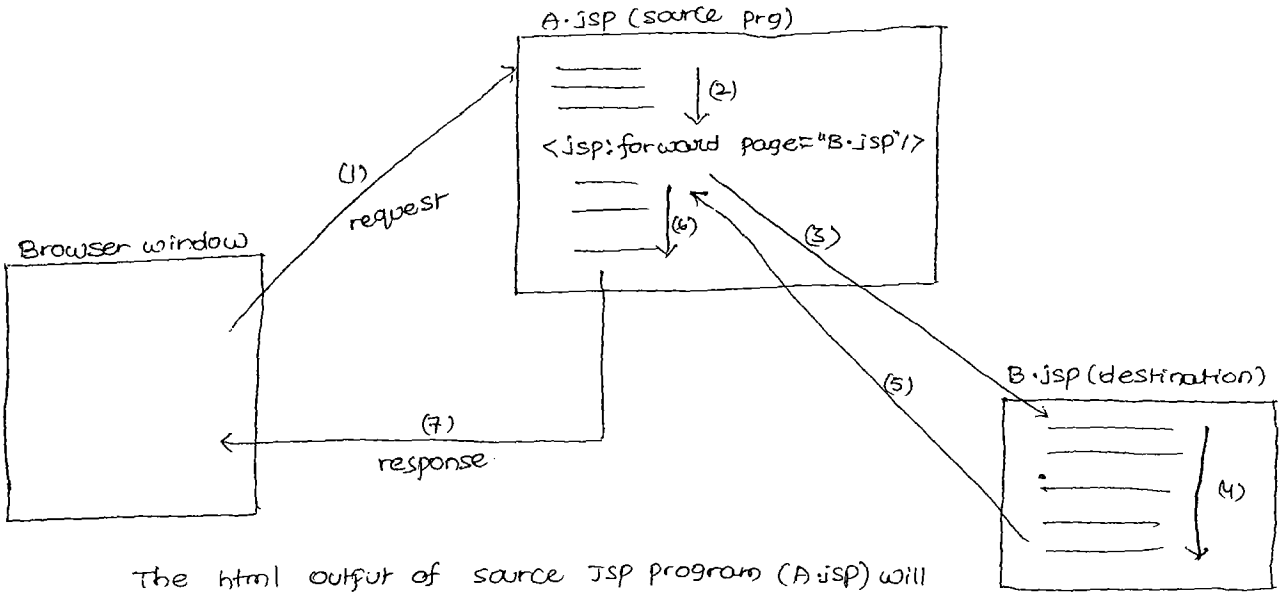
* For summary table that talks about all JSP tags refer page nos 112 & 113.

<jsp:forward>

Syntax

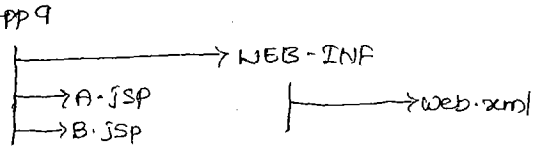
```
<jsp:forward page="dsturl"/>
```

* Given to perform forwarding request mode of JSP chaining. It internally uses rd.forward(-,-) method.



The html output of source JSP program (A.jsp) will be discarded and only the html output of (B.jsp) (destination prg) goes to browser window as response through A.jsp (src jsp prg). Here two seperate and Independent equalent servlet programs will be generated for A.jsp, B.jsp programs.

Example application



A.jsp

```
<b> start of A.jsp </b> <br>
<jsp:forward page="B.jsp"/>
<b> end of A.jsp </b>
```

B.jsp

```
<br><%=new java.util.Date()%> <br>
<i> from B.jsp </i>
```

* When source jsp program is interacting with destination jsp program either by using <jsp:forward> or <jsp:include> then source jsp program can send additional request parameter values to destination JSP program by using <jsp:param> tag.

↓

This tag is allowed to use only as sub tag of <jsp:include>, <jsp:forward> tags.

Example

In A.jsp of JspApp9

```
<b> start of A.jsp </b> <br>
<jsp:forward page="B.jsp">
    <jsp:param name="bookname" value="CRJ"/>
    <jsp:param name="price" value="300"/>
</jsp:forward>
<b> end of jsp </b>
```

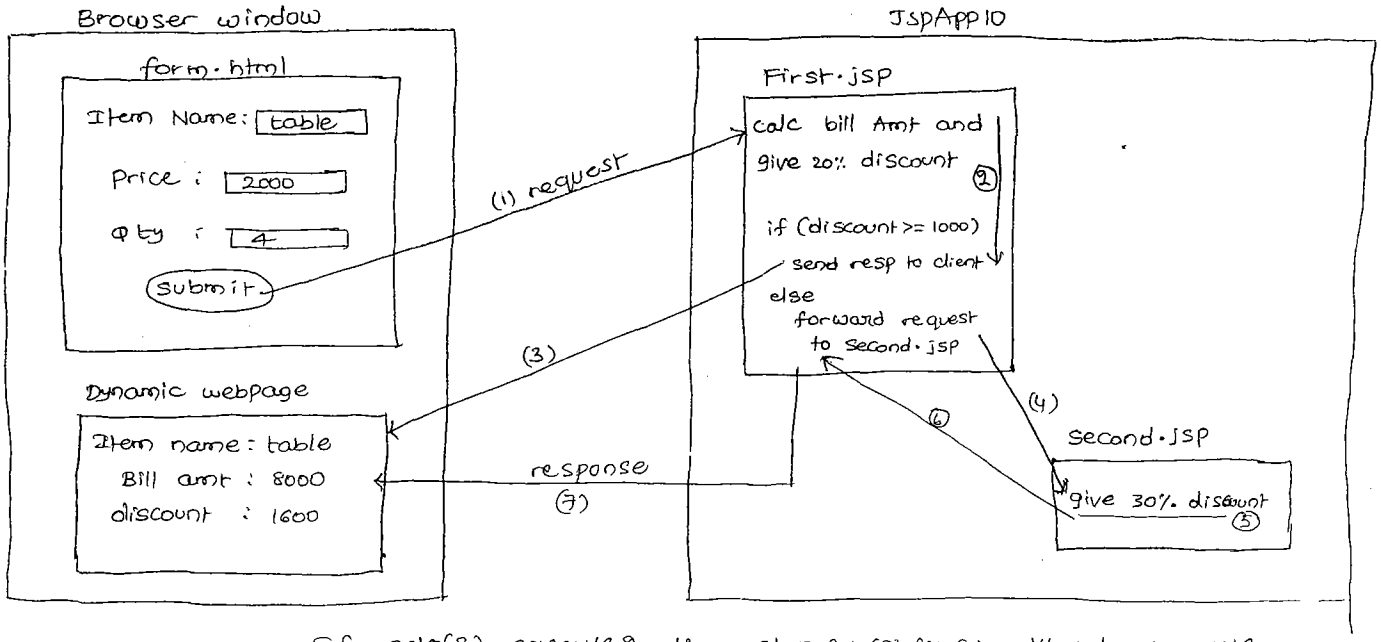
} Additional request parameters name, values

In B.jsp of JspApp9

```
<br><i> from B.jsp </i>
<br> Additional request parameters given by A.jsp are
<% = request.getParameter("bookname") %>
<% = request.getParameter("price") %>
```

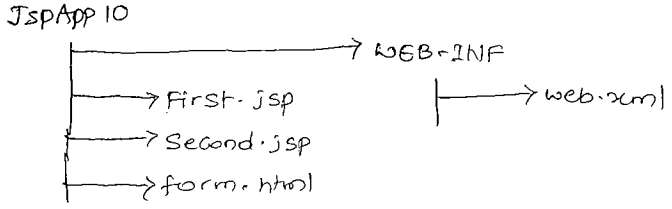
} logic to read additional request parameter values

Example application on <jsp:forward>, <jsp:param> tags



If step(3) executes then step (4), (5), (6), (7) will not execute.

Deployment directory structure



Procedure to develop above application by using MyEclipse IDE

NOTE: It is always recommended to execute the <jsp:forward> tag in source jsp program by enabling the Condition....

Step(1): Launch MyEclipse IDE by choosing workspace folder.

Step(2): create web project in MyEclipse IDE.

File menu → new → web project → JspApp10 → Finish

Step(3): Add form.html to webroot folder of the project.

Right-click on webroot folder → new → JSP html → file name: form.h
→ finish

form.html

```
<form action="First.jsp" method="get">
  Item name: <input type="text" name="tname" /> <br>
  Item Price: <input type="text" name="tprice" /> <br>
  Item qty: <input type="text name="tqty" /> <br>
  <input type="submit" value="submit" />
</form>
```

Step(4): Add First.jsp to the webroot folder of the Project.

Right click on webroot folder → new → JSP → File name: First.jsp → Finis

First.jsp

```
<% //read form data
   String iname = request.getParameter("tname");
   int price = Integer.parseInt("tprice");
   int qty = Integer.parseInt(request.getParameter("tqty"));
   // calculate bill amount
   int bamt = prince * qty;
   // give 20% discount
   float discount = bamt * 0.2f;
   if (discount >= 1000)
     {
       out.println("Item name:" +iname);
       out.println("Bill Amt:" +bamt);
       out.println("Discount:" +discount);
     }
   else
     {
       // forward the request to second.jsp %>
```

```
<jsp:forward page="second.jsp">
    <jsp:param name="p1" value="<%= iname %>" />
    <jsp:param name="p2" value="<%= bamt %>" />
</jsp:forward>
<%@%>
```

} Sending iname, bamt details to second.jsp from first.jsp as additional request parameter value

Step(5): Add second.jsp to webroot folder of project.

second.jsp

```
<b> from second.JSP </b><br>
<% // read additional request parameters sent by First.jsp
    String name = request.getParameter("p1");
    int bamt = Integer.parseInt(request.getParameter("p2"));
    //calculate 30% discount amount
    float discount = bamt * 0.3f;
    //display item details
    out.println("Item name:<br>" +name);
    out.println("Bill amt:<br>" +bamt);
    out.println("Discount:<br>" +discount); %>
```

Step(6): Configure Tomcat Server with MyEclipse IDE.

window menu → preferences → MyEclipse → Servers → Tomcat → Tomcat 6.x → Enable → <Tomcat_home> directory → apply → ok.

Step(7): Start Tomcat server

Goto Servers icon in the tool bar → Tomcat 6.x → Start

Step(8): Deploy the project in Tomcat server.

Go to deploy icon in the tool bar → project → IspApp10 → add → server : Tomcat 6.x → Finish → Ok

Step(9): Test the application

Open browserwindow & type this URL

http://localhost:2020/JspApp10/form.html

There are directive include, Action include tags. can you tell me why there is no directive forward tag? Only Action forward tag is available in JSP

(A) The basic work of forwarding request operation is discarding the output of source JSP program.

The Directive tags perform their activity at translation phase by including the code of destination program to the code of JSP equalent servlet generated for source JSP program.

In this process discarding the output of source JSP program is impossible. so there is no directive forward tag in JSP.

What is the difference between <jsp:forward> tag and response.sendRedirect(-) method?

(A)

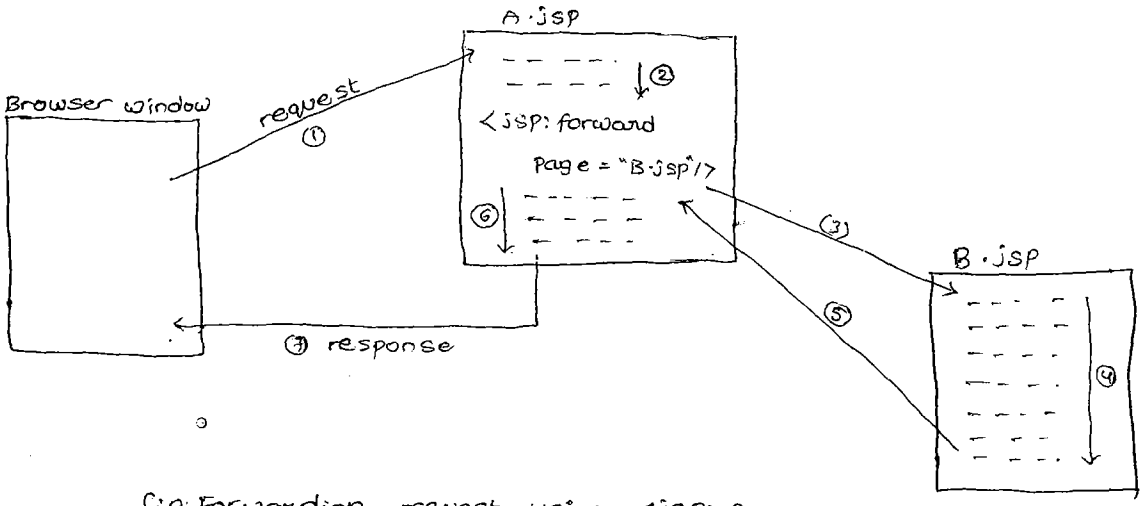


fig: Forwarding request using <jsp:forward> tag

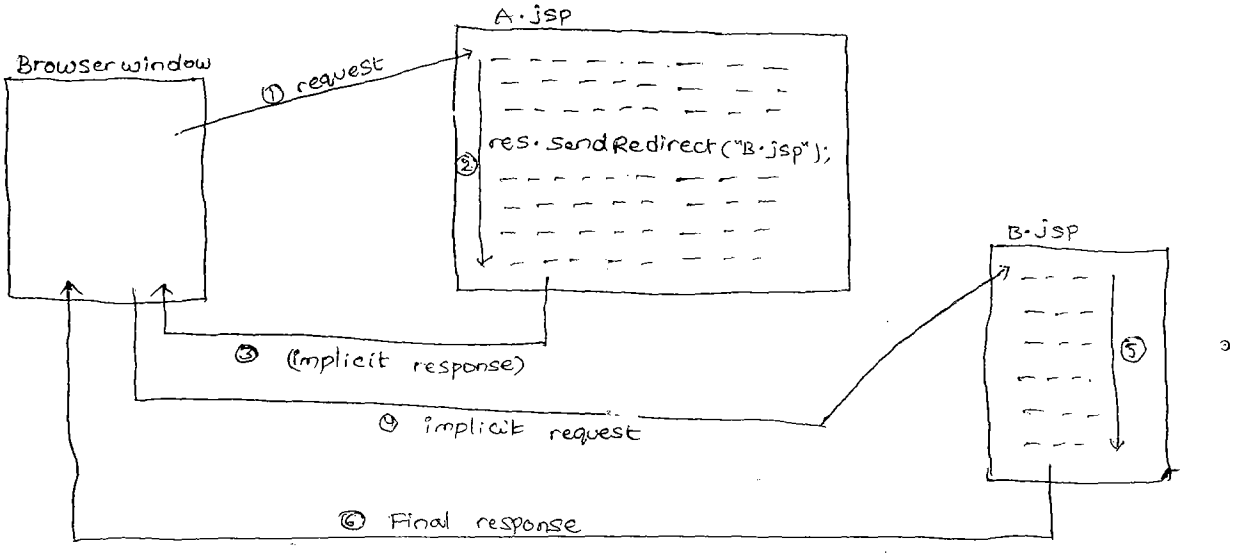


fig: sendRedirection using res.sendRedirect(-) method.

<jsp:forward>	response.sendRedirect(-)
* source JSP program communicates with destination program directly.	* Source Jsp program communicates with destination program by having one network round trip with browser window
* Both source, destination programs use same request and response objects so request data coming to source JSP program can be used in destination program.	* source JSP program and destination program will not use same request and response objects so the request data coming to source destination program is not visible and accessable in destination program.
* Both source and destination programs must be there in same web application.	* Both source and destination programs can be there in same web application or can be there in two different web applications of same server or different servers.
* A destination program must be a html program or JSP program or servlet program.	* Destination program can be html program or java based web resource program or non-java based web resource program.
* while forwarding request url in the address bar will not be changed.	* while redirecting the request the URL in the address bar will be changed into destination program related URL.

* <jsp:forward>, <jsp:include> tags are the tag based alternates for rd.forward(-,-), rd.include(-,-) methods. and there is no JSP tag providing the functionality of response.sendRedirect(-) method. That means java code should be written directly in JSP program.

What is the difference between the implicit objects page & pageContext?

(A) The implicit object page holds "this" key word which is nothing but reference of currently invoking JSP equalent servlet class object.

→ pageContext object holds multiple details of JSP page like this, request, response objects, errorPage name, buffer size, auto flush mode, session objects availability status.

→ JSP equalent servlet creates / gets Access to application, Config, ses. and out implicit objects through this PageContext object.

Code in JSP equalent servlet program

for implicit object page

```
object page = this;
```

for implicit object pageContext

```
pageContext = _jspxFactory.getPageContext(this, request, response,
                "Abc.jsp", true, 10240, true);
```

"Abc.jsp" → error page; true → availability of status of session object; 10240 → buffer size; true → auto flush mode

How to pass data between the JSP programs of web application?

(A) (1) If source and destination programs are using same req, res objects

(a) use request attributes

(b) use additional request parameters (<jsp:param>)

(2) If source and destination JSP programs are getting request from same browser window and not using same req and res objects

use session attributes

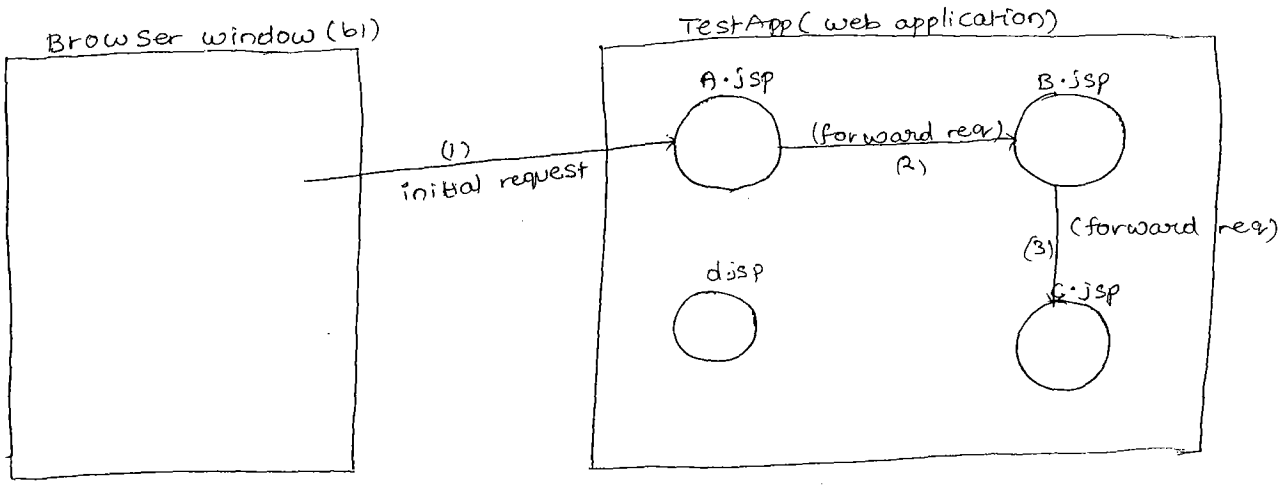
(3) If source and destination programs are not getting request from same browser window and not using same request and response objects

use application attributes.

(4) Use pageContext attributes

we can create pageContext attribute having pageScope (or) request Scope (or) session scope (or) application scope.

— o —

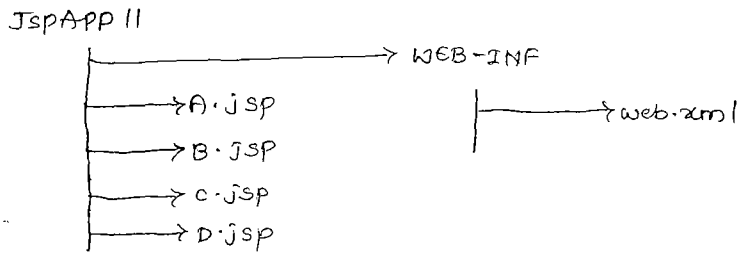


* In the above diagram the request attribute created in A.jsp is visible and accessable in B.jsp, C.jsp programs because A.jsp, B.jsp, and C.jsp programs use same request, response objects.
* Request attributes are visible through out request cycle.
* The session attribute created in A.jsp by getting request from browserwindow b1 is visible and accessable in all jsp programs of web application irrespective their request, response objects but all these web resource programs must get request from same browser window b1.

NOTE: session attributes are visible in all web resource programs of web application but they are specific to a browser window.

* Application attribute created in A.jsp is visible and accessable in all other web resource programs of web application irrespective of the browser window from which they are getting request and irrespective of request and response objects they are using.

Example application



Attributes are logical names holding values the servlet programming, JSP programming uses attributes to pass data from one web resource program to another web resource program

setAttribute(-,-) → To create new attribute or to modify existing attribute values

getAttribute(-,-) → To read existing attribute values

removeAttribute(-,-) → To remove existing attribute

A.jsp

```
<% //create attributes
    request.setAttribute("attr1", "val1");
    request.setAttribute("attr2", "val2");
    application.setAttribute("attr3", "val3");

    <!-- forward the request to B.jsp -->
    <jsp:forward page="B.jsp" />
```

B.jsp

```
<!-- read attribute values --> <b> from B.jsp </b> <br>
attr1 (req) attribute value is = <%= request.getAttribute("attr1") %> <br>
attr2 (session) attribute value is = <%= session.getAttribute("attr2") %>
attr3 (application) attribute value is = <%= application.getAttribute("attr3") %>
<!-- forward the request to C.jsp -->
<jsp:forward page="C.jsp" />
```

C.jsp

same as B.jsp but don't forward request any where

D.jsp

same as C.jsp

URL to test the application

give first request to A.jsp and give other request to other jsp progra either from same browser window or from different browser windows.

* Request object can create only request scope attributes.
* session object can create only session scope attributes.
* application object can create only application scope attributes. where as pageCon object can create page scope or request scope or session scope or application scope attributes.

To create pageContext attribute

```
pageContext.setAttribute("attr1","val1"); → create pageContext attribute having page scop
pageContext.setAttribute("attr2","val2", pageContext.SESSION_SCOPE);
```

→ create pageContext attribute having page/scope: session scope.

other possible scopes

pageContext.PAGE_SCOPE

pageContext.SESSION_SCOPE

pageContext.APPLICATION_SCOPE

pageContext.REQUEST_SCOPE

All these scopes constants of javax.servlet.jsp.PageContext class.

TO ~~mod~~modify pageContext attribute values

```
PageContext.setAttribute("attr1", "val1");
```

→modifies attr1 value of page scope

```
pageContext.setAttribute("attr2", "val122", PageContext.SESSION_SCOPE);
```

→modifies attr2 attribute value of session scope.

To read pageContext attribute values

```
String s1 = (String) pageContext.getAttribute("attr1");
```

→Reads attr1 attribute value from page scope.

```
String s2 = (String) pageContext.getAttribute("attr2", PageContext.SESSION_SCOPE);
```

→Reads attr2 attribute value from session scope.

To remove pageContext attributes

```
page Context.removeAttribute("attr1");
```

→remove attr1 value attribute from page.

```
pageContext.removeAttribute("attr2", pageContext.SESSION_SCOPE);
```

→removes attr2 attribute from session scope.

To find attribute

```
String s1 = (String) pageContext.findAttribute("attr2");
```

→searches and reads attr2 attribute value in multiple scopes in the following order...

a) page scope b) request scope c) application scope d) session scope.

What is the difference between getAttribute() and findAttribute() invoked on the pageContext object?

(B) getAttribute() searches for the given attribute only in the specified scope (no scope is specified page scope) where as findAttribute() searches for the given attribute if multiple scopes in a particular order like page scope, request scope, session scope, application scope.

Revised JspApp11 application with the support of pageContext attribute

A.jsp

```
<% // create attributes
   pageContext.setAttribute("attr1", "val1", pageContext.REQUEST_SCOPE);
   pageContext.setAttribute("attr2", "val2", pageContext.SESSION_SCOPE);
   pageContext.setAttribute("attr3", "val3", pageContext.application_SCOPE);
   <!-- forward the request to B.jsp -->
    <jsp:forward page="B.jsp" />
```

B.jsp

```
<!-- //read attribute values -->
  <b> from B.jsp </b>
  attr1 (req) attribute value is = <%=pageContext.getAttribute("attr1", pageCo
                                    REQUEST_SCOPE) %><br>
  attr2 (session) attribute value is = <%=pageContext.findAttribute("attr2") %>
  attr3 (application) attribute value is = <%=pageContext.findAttribute("attr3") %>
  <!-- forward the request to C.jsp -->
   <jsp:forward page="C.jsp" />
```

C.jsp

Same as B.jsp but don't forward request any where.

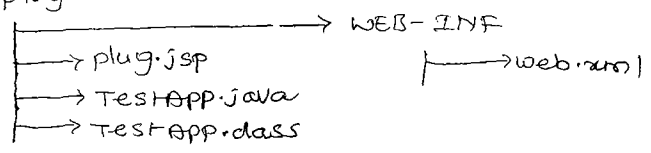
D.jsp

Same as C.jsp

* <jsp:plugin> tag is given to display applets on the browser window. This tag internally uses <applet> tag or <object> tag of html to display applets.
* <jsp:fallback> tag should be used as sub tag of <jsp:plugin> tag. This tag executes only when underlaying browser window does not support applets.
* The Tomcat web server internally uses <embed>, <noembed> tags for <jsp:plugin> <jsp:fallback> tags.

Example application

jsp-plug



Request URL to test the application

http://localhost:2020/jsp-plug/plug.jsp

TestApp.java

```
import java.applet.Applet;
import java.awt.*;
public class TestApp extends Applet
 {
      public void paint (Graphics g)
       {
          setBackground(color.red);
          g.drawString("this is from applet to test", 50, 50);
       }
 }
```

plug.jsp

```
<html>
<body bgColor="pink">
   <jsp:plugin type="applet" code="TestApp.class" codebase="/jsp-plug" (*1) width="300"
                                                                        height="300" >
      <jsp:fallback> browser does not support applets </jsp:fallback>
   </jsp:plugin>
</body>
</html>
```

web.xml

```
   <web-app/>
```

(*1) → specifies the directory where given applet class is available.
(TestApp.class)

Java Bean :

* A java bean is a java class that contains getter and setter methods. There is no special key word to develop the java class as java bean.
* The member variables of java bean class are technically called as bean properties and every bean property contains one getter and one setter method.
* Getter methods are useful to get or read the data from bean properties. where as setter methods are useful to write the data to bean properties.

Ex:

```
public class StudentBean
  {
```

```
//bean properties
  private int sno;
  private String sname;
  private float avg;
// write getXxx() and setXxx(-) methods.
  public void setSno (int no)
    {
      this.no=no;
    }
  public int getNo( )
    {
      return no;
    }
  public void setSname (String sname)
    {
      this.sname=sname;
    }
  public String getSname( )
    {
      return sname;
    }
  public void setAvg (float avg)
    {
      this.avg=avg;
    }
  public float getAvg ( )
    {
      return avg;
    }
}//class.
```

* If java programmers follow java bean standards while developing their java classes then it becomes easy for the programmers to exchange the java classes.
* Java bean is a standard to develope java classes having getXxx() and setXxx(-) methods.
* the best way to develop re-usuable simple java class is developing that class as java bean class.

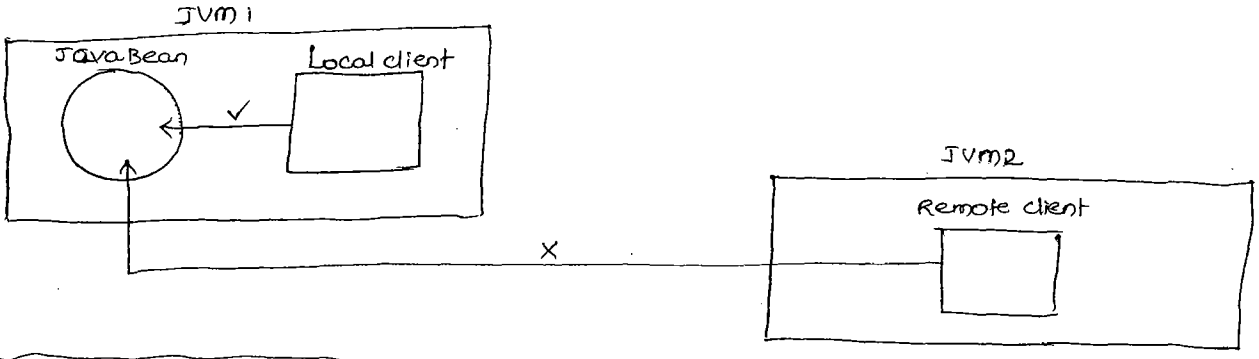
what is the difference between java bean and EJB (Enterprise Java Bean)?

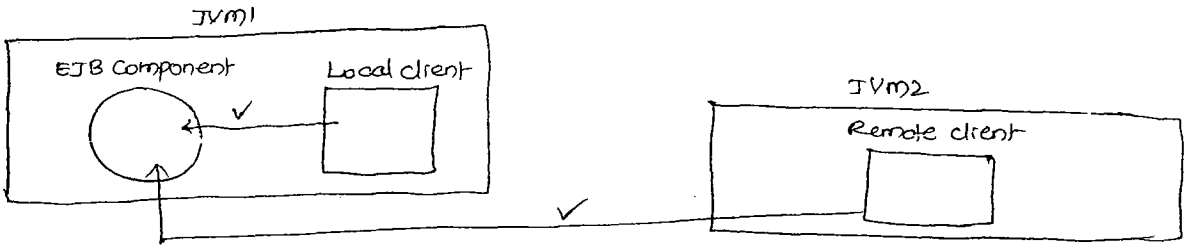
Java bean	EJB
* An ordinary java class containing getter and setter methods.	* Distributed technology develop distribut applications.
* JUST needs JVM for execution	* Needs JVM and EJB Container for execution.

* No life cycle methods.
* useful as a reusable helper class in projects
* Allows only local clients.

* Life cycle methods are there.
* useful as technology to develop business components of the project.
* Allows both local and remote clients.

* In one computer multiple JVMs can be there to run multiple java applications simultaneously or parallelly. If application and its client resides in the same JVM then that client is called local client to application. If application and its client resides on two different JVMs of same computer or different computers then that client is called remote client to application





* A Java Bean class can have 3 types of Bean properties.

(1) Simple Bean properties
(2) boolean Bean properties
(3) Indexed Bean properties

Simple Bean properties

* Allows to store only one value at a time.

Ex:
```
int sno;
public void setSno(int sno)
    { this.sno=sno;
    }
public get getSno()
    { return sno;
    }
```

Boolean Bean properties

* Allows to store only true or false value.

Ex: boolean married;

```
public void setMarried(boolean married)
        {
          this.married = married;
        }
public boolean getMarried()/isMarried()
        {
           return married;
        }
```

Indexed Bean properties

* Allows to store multiple values. This bean property type is an array.

Ex: String[] colors;

```
public void setColors(String[] colors)
    {
       this.colors = colors;
    }
public String[] getColors()
     {
       return colors;
     }
```

* For servlet program to javaBean communication the servlet program should be an object of javabean class and should call methods on that object.
* For jsp program to javabean communication we can use the following tags

```
<jsp:useBean>
<jsp:setProperty>
<jsp:getProperty>
```

java based web application development models

1. Model 1 Architecture

→ use only servlets or only JSP programs as server side web resource programs of web application

2. Model 2 architecture (MVC Architecture)

1. MVC1 architecture
2. MVC2 architecture

→ In MVC1 and MVC2 architectures multiple technologies support will be taken to develop web applications.

Logics that can be there in our java web applications

(1) Request Data gathering logic

(2) Form validation logic

(3) Business logic

(4) session management logic

(5) persistance logic

(6) middlewaore services logic

(7) Presentation logic and etc...

* Request Data gathering logic is responsible to gather various details given by client in its httprequest like form data, header values and etc. For this, use getXxx() methods on request object.
* In form validation logic checks the format and pattern of the form data. Ex: checking wheter email id is having @, . symbols or not.
* Business logic processes the request and generates the results.
* Business logic is main logic of web application.
* session management logic remembers the client data across the multiple requests during a session through session tracking techniques like hidden form fields, cookies, http session with cookies and Http session with URL rewriting and etc...

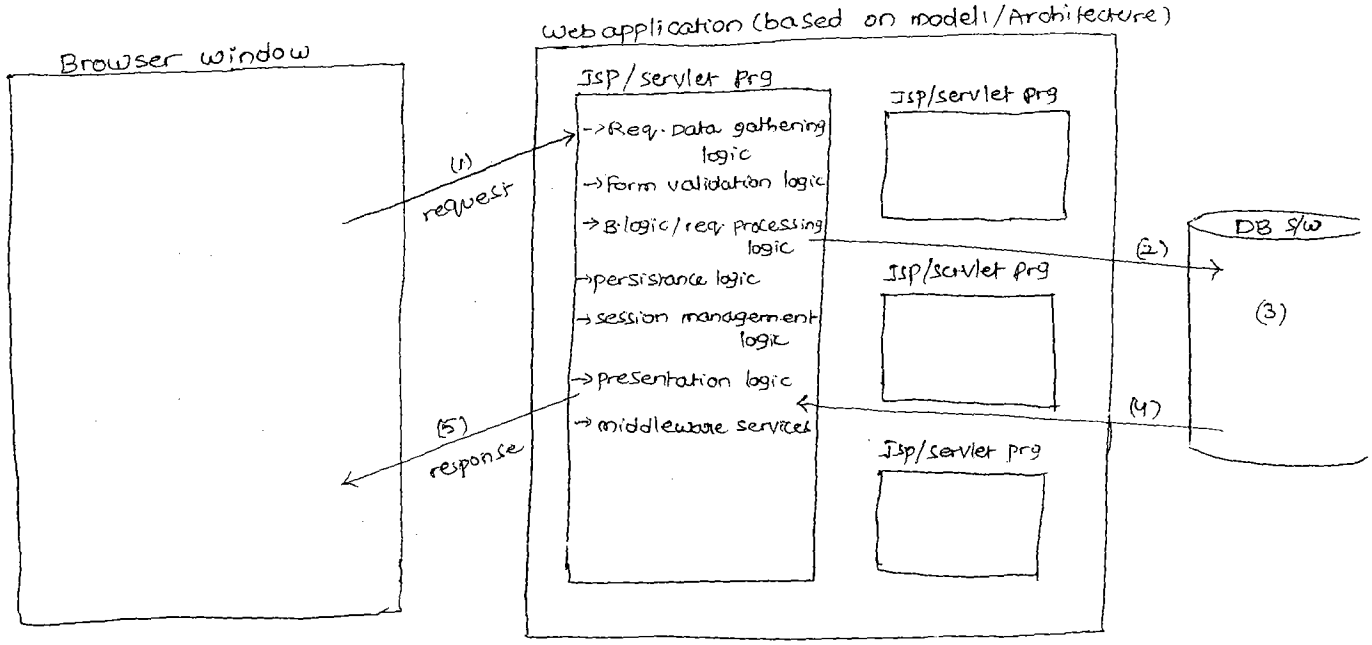
NOTE: There are no special tracking techniques in JSP. JSP should also use the above said four session tracking techniques.

* persistance logic is responsible to manipulate database data by performing "curd" operations. (Ex: JDBC code)
* middleware services are the additional services which can be applied on the applications like JDBC connection pooling, security and etc...
* Presentation logic generates user interface required for end user. It is also responsible to format the results and to display the results. (html code).

Model-1 architecture

* In this model either servlet programs or jsp programs will be used as server side web resource programs. more ever if servlet programs are used the JSP programs will not be used and viceversa.

* In every server side program multiple logics will be mixed up.



* In model1 architecture web application, the client side web resource programs c be there but either servlets or jsp must be used as server side programs
* Most of the programmers prefer working with JSPs (compare to servlets) while developing mod architecture base web application.

* Disadvantages of model1 architecture

* multiple logics will be mixed up in every server side web resource program. This indicates there is no clean seperation of logics and this also indicates mod fication done in one logic may disturb other logic.
* Parallel development is not possible so the productivity is very poor.

NOTE: Doing more work in less time with good accuracy gives good productivi

* Middleware services (some) must be implemented by the programmer manually This improves burden on the programmer.

Advantages

* knowledge on either servlet or JSP is sufficient to develop web resource progra
* since parallel development is not possible multiple programmers are not required to develop the web applications.

* TO overcome the drawbacks of model-1 architecture use model-2 (or MVC architectures to develop the web applications.

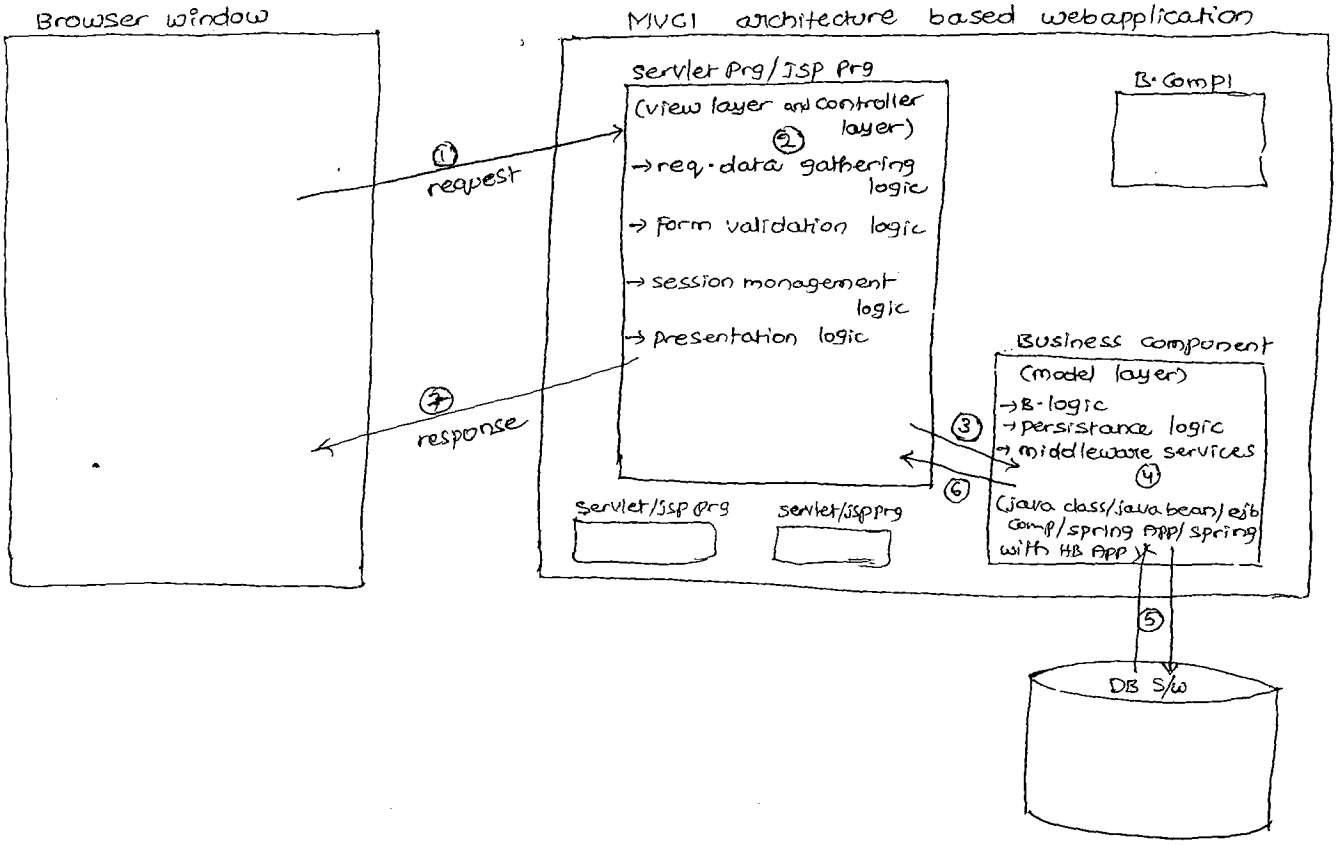
MVC/ model-2 architecture

M ⟶ model Layer → Represents b·logic + persistance logic
→ use java class/ java bean/ ejb components/ spring App / spring with HB App in model layer to develop logics.
→ It is like Accounts officer.

V ⟶ View Layer ⟶ Represents presentation logic
⟶ use html/ jsp programs to develop this presentation logic.
→ It is like Beautician.

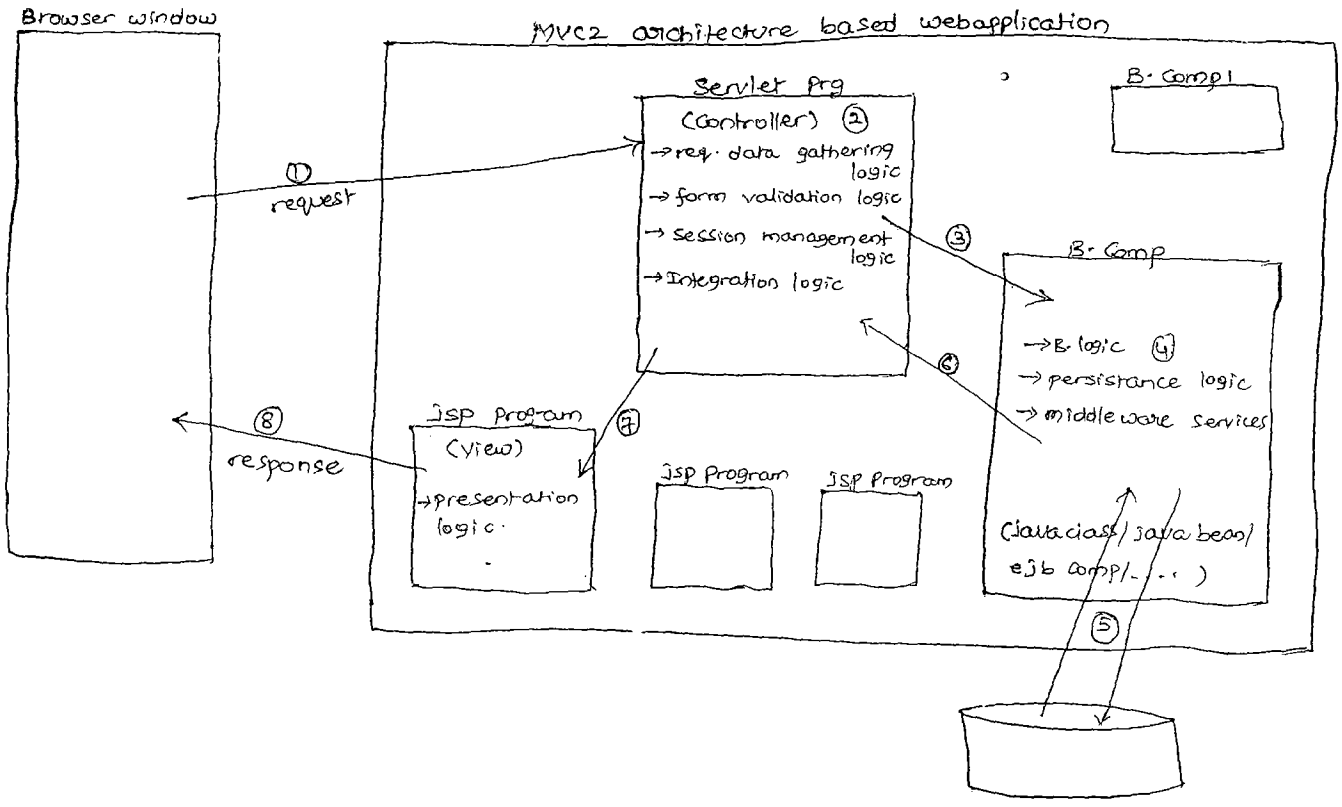
C ⟶ Controller Layer → Represents Integration logic/ Connectivity logic
→ use servlet program or servlet filter program to develop these logics.
→ It is like Traffic police/ supervisor

* Integration logic is responsible to take the request from browser window, to pass that request to model layer resource, to gather result from model layer resource and to pass the result to view layer resource.
* Integration logic is responsible to control and monitor every operation of the web applications execution.
* In model 2/ mvc architecture based web application development multiple technologies support will be taken by programmers.
* In model mvc1 architecture the same servlet programs/ JSP programs will act as view layer, Controller layer resources representing presentation, business logics and seperate resources will taken for model layer.
* In mvc2 architecture three different resources will be taken in 3 different layers.
Jsp programs ⟶ In view layer
servlet program ⟶ In Controller layer
java class/ javabean/ ejb component/... ⟶ In model layer
* In mvc1 architecture multiple resources can be there in each layer.
* In mvc2 architecture multiple resources can be there in view, model layers but only one servlet program will be in Controller layer.



* MVC1 gives quite better clean seperation of logics compare to model-1 becau it is seperating business logic and persistance logic from servlet program/ JSP program. But for more clean seperation of logics it is recommended to use MVC2 architecture.

MVC-2 architecture



with respect to the diagram

① Browser window gives request to web application.

② A/S a controller servlet program traps and takes the request

③ Servlet program passes the request to model layer business component by using integration logic.

④, ⑤ → The business logic of business component process the request and generates the results. Also ~~talk~~ talks with DB s/w if necessary.

⑥ → Business Component generated results goes back to servlet program (Controller)

⑦ The controller servlet program passes the results to the view layer Jsp program

⑧ The Jsp program uses the presentation logic to format the results and sends formatted result to browser window as response.

* use model-1 architecture to develop small scale web application.
* use mvc-1 architecture to develop medium scale web application.
* use mvc-2 architecture to develop large scale web application.

Advantages of MVC-2 architecture

(1) There are multiple layers in web application development so we can acheive clean separation of logics.

(2) modifications done in logics of one layer does not effect logics of another layer.

(3) Parallel development is possible so productivity will be good.

(4) Provides easyness in enhancement and maintanance of the web application.

(5) MVC-2 is a ~~s~~ industry standard to develop the web applications.

(6) When spring, hibernate, EJB technologies are used in the model layer we can get the benifit from middleware services (built-in). This reduces burden on the programmer.

(7) MVC-2 is industry standard to develop the web applications.

disadvantages

* For parallel development multiple programmers are required.
* knowledge in multiple technologies is required to develop the web applications.

While we developing MVC2 architecture based web applications along with taking support from multiple technologies we should also implement set of rules which are called MVC-2 rules or principles.

(1) Every layer is given to have certain logics. Just place only those logics and don't place any excess logics.

(2) It is recommended to use JSPs use in view layer, servlets in controller l use javaBean or java class in model layer when business logic is less.

(3) use EJB or spring with HB in model layer when business logic is huge.

(4) Every operation in the web application must take place under the Controller c Controller servlet.

(5) There can be multiple resources in the view layer and multiple resources in model layer but there must be only one controller servlet or controller servlet Filter program in Controller layer.

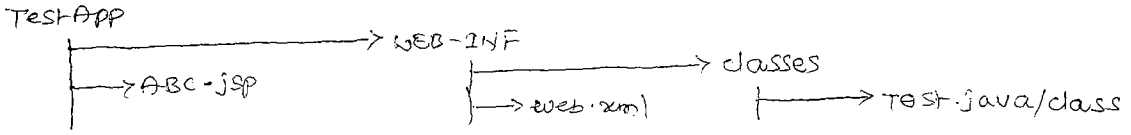
(6) The view layer resources must not talk with model layer resources direct and vice versa. That means they must communicate with each other through Controller servlet.

(7) Two resaures of view layer (JSP programs) must not talk with each other directly. That mean they must communicate with each other through Controller layer.

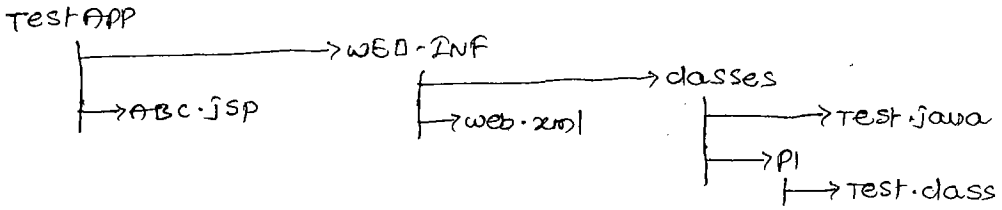
* In a web application where jsp to java bean communication is taking plac with the support of <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> tags comes under MVC-1 architecture based web application development.

* A java bean class can also Contain business methods along with getter an setter methods.

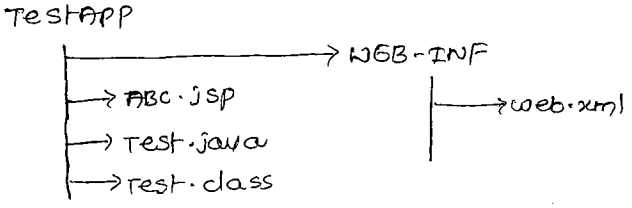
* If you want to use java class or java bean in Jsp program then it mus be placed in a package defined in WEB-INF\classes folder. Otherwise it must be placed along with Jsp program. If that java class is directly there in WEB-INF\classes folder then it can not be used in Jsp progr



Test class can not be used in ABC.jsp



Test class can't be used in ABC.jsp.



Test class can be used in AB.jsp.

<jsp:useBean>

Syntax : <jsp:useBean attributes/>

used for creating/locating an object of javabean class.

attributes

id = " " // instance name (object name)

class = " " // a fully qualified javabean class name.

scope = "page/session/request/application" (default scope is "page")

// specifies the scope of the bean class obj

page (default)

bean class obj is available through out the request page.

session

bean class obj is available through out the request session

request

throughout the request

application

available for all pages within the context (web application)

beanName = "logical name of java bean"

type = "reference class type of java bean class object".

Ex: <jsp:useBean id="st" class="p1.StudentBean" scope="session" />

creates p1.StudentBean class obj having name "st" and keeps in session scope.

If this object is already available in session scope then it locates that object from session scope. For this it internally uses pageContext.setAttribut pageContext.getAttribute() methods.

The above statement creates object like this:

pl.StudenBean st = new pl.StudentBean();

"st" object type is pl.StudentBean

"st" reference object type is pl.StudentBean

Ex:

<jsp:useBean id="st" type="ABC" class="pl.StudentBean" scope="session" />

creates or locates pl.StudentBean class object (st) in from request scope.

Bean class object creation statement

ABC st = new pl.StudentBean();

"st" → reference type is ABC class

"st" → object type is pl.StudentBean class

→ Here pl.StudentBean class extends from ABC class.

* <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> are 3 independent tags

<jsp:setProperty>

calls setXxx methods on java class object to set given values to bean properties.

Syntax: <jsp:setProperty attributes />

attributes

property = " " // bean property name

name = " " // (bean class obj name) id attribute value given in the <jsp:use

value = " " // value to be set for bean property

param = " " // input (request) parameter name

NOTE: value or param → any one attribute has to be used at a time.

Ex: <jsp:setProperty name="st" property="sno" value="567" />

// this internally calls setSno() methods on st obj and assigns value "567" to the bean property "sno".

Ex: <jsp:setProperty name="st" property="sname" param="stname"/>

this internally calls setSname(-) method on st obj to assign the value of "stname" req param to sname bean property.

<jsp:getProperty>

calls getXxx() methods to read values from bean properties.

syntax: <jsp:getProperty attributes/>

attributes

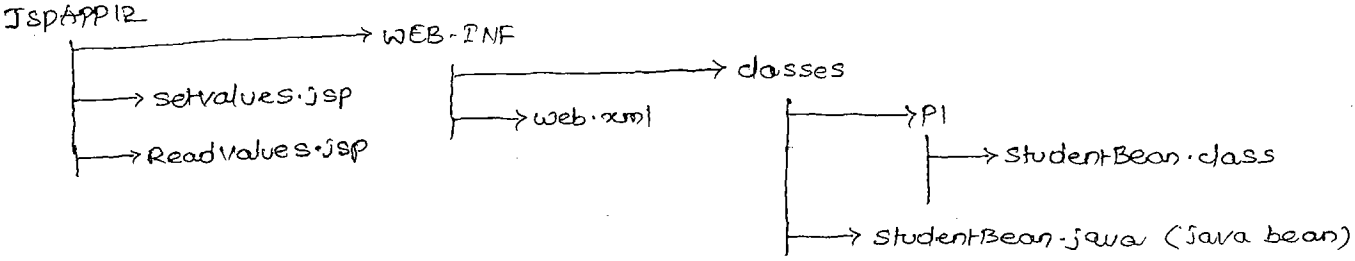
name="" // (bean obj name) id attr value given in <jsp:useBean>

property="bean property name"

ex: <jsp:getProperty name="st" Property="sno"/>

gives "sno" bean property value by calling getSno() method on bean class object ("st") and also displays that value on browser window.

Example application on Jsp to Java bean Communication (MVC-1 architecture)



StudentBean.java

```
package P1;
public class StudentBean
 {
   // bean properties
     private int sno;
     Private String sname;
     private float avg;
     // Constructor
     public StudentBean()
      {
        System.out.println("StudentBean 0-param constructor");
      }
       // write getXxx() and setXxx() methods
```

```
    public void setSno(int sno)
    {
      s.o.p("StudentBean: setSno(-) method");
      this.sno = sno;
    }
    ========
  }
```

> javac -d . StudentBean.java

SetValues.jsp

```
<%@ page import="p1.StudentBean" %>
<!-- create or locates java bean class obj -->
<jsp:useBean id="st" class="p1.StudentBean" scope="session" />
<!-- set values to bean properties -->
<jsp:setProperty name="st" Property="sno" value="101" />
<jsp:setProperty name="st" Property="sname" value="rajesh" />
<jsp:setProperty name="st" Property="avg" value="67.89" /> <br> values are set to bean prope
```

ReadValues.jsp

```
<%@ page import="p1.StudentBean" %>
<!-- create or locate javabean class obj -->
<jsp:useBean id="st" class="p1.StudentBean" scope="session" />
<!-- read and display bean properties values on browser window -->
  sno value is : <jsp:getProperty name="st" property="sno" /> <br>
  sname value is : <jsp:getProperty name="st" property="sname" /> <br>
  avg value is : <jsp:getProperty name="st" property="avg" />
```

web.xml

```
<web-app/>
```

* Irrespective of whether Thread safety is enabled or not enabled (isTh
Safe="true"/"false") the jsp equalent servlet manages session scoped, app
cation scoped javabean class objects having thread safety with the suppor
of synchronised blocks.

URL to test the application.

http://localhost:2020/JSPApp12/setValues.jsp

http://localhost:2020/ReadValues.jsp.

* we can use the param attribute of <jsp:setProperty> tag to set the received request parameter value as bean property value.

```
<jsp:setProperty name="st" property="sno" param="stno"/>
<jsp:setProperty name="st" property="sname" param="stname"/>
<jsp:setProperty name="st" property="avg" param="stavg"/>
<br> values are set to bean properties
```

property → bean property name; param → req. param name

Request URL is

http://localhost:2020/JspApp12/setValues.jsp?stno=3456&stname=ramesh&stavg=37.5

query string representing req. params

* If request parameter names are matching with bean property names then we can use property = "*" in <jsp:setProperty> tag to set request parameter values as bean property values.

setValues.jsp

```
<%@ page import="p1.StudentBean" %>
<jsp:useBean id="st" class="p1.StudentBean" scope="application"/>
<jsp:setProperty name="st" property="*"/>
<br> value are set to bean properties.
```

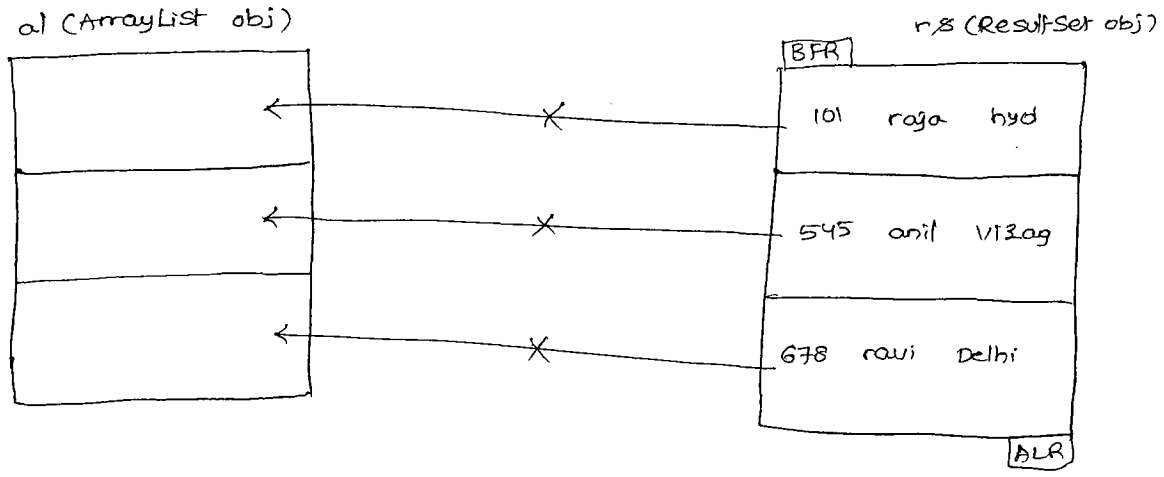
* The application that can display and rotate advertisements / Ads in web pages is called as AdRotator.
* Our AdRotator application is having the following features.
  (1) Should Render the advertisments as graphical hyperlinks pointing to client's related web sites.
  (2) The advertisments must be changed automatically at regular intervals.
  (3) Advertisment should be rendered randomly in the web page.
  (4) Develop application based on MVC1 architecture.
* For example application on advertisment rotator refer application - 6 of page nos 110 and 111.

Mini Projects Discussion

* In order to send java objects over the network it must be developed serializable object.
* Java object becomes serializable object only when its class implements java.io.Serializable interface directly or indirectly.
* All Collection framework data structure objects are serializable objects by default.
* ResultSet object is not serializable object so we can not send that obje over the network. But in multilayer environment (like mvc) we need to send the Res Set object data over the network.
* To solve the above said ResultSet Problem there are two solutions.

  solution(1): use RowSets as alternate for ResultSets

  solution(2): Copy the data of ResultSet obj to Collection Framework datastruc and send them over the network.
* RowSets are serializable objects so we can send them over the network b very few jdbc drivers are support for RowSets. Due to this industry prefer working with solution(2) to send ResultSet object data over the network.
* If you are performing only read operations on data structures after receving them then it is recommended to work with non synchronized data structures like ArrayList, HashMap for better performance.
* If you want to perform read write and other operations on the received Collection Framework data structure then it is recommended to work with synchronized data structure like vector, Hashtable to avoid multi threading related concurrency issues.
* we can not move ResultSet object records directly to ArrayList object elements on each record to each element basis because in one elemen of ArrayList object we can store only one java object at a time. Bu in each record of ResultSet there is a possibility of having multiple objects and other values.



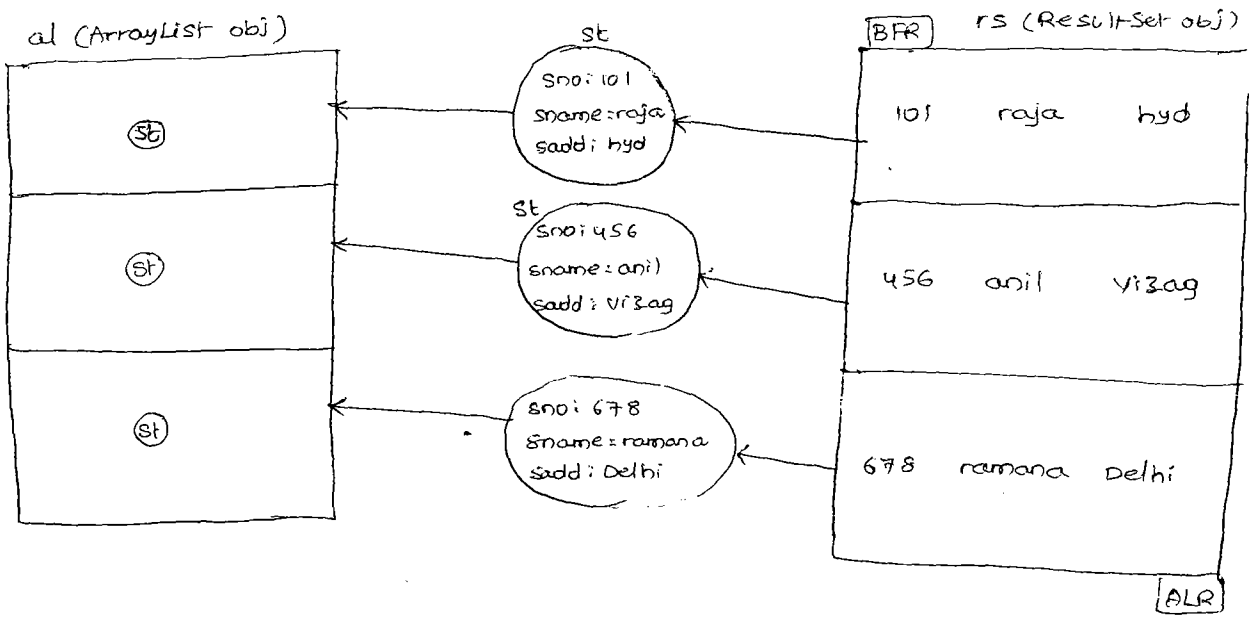
* To solve the above problem store each record values of ResultSet in a user defined java class object and add that object to each element of ArrayList object. Here this user defined java class is a java Bean and it is technically called as DTO class or VO class (value Object).

D.T.O class or V.O class

```
public class StudentBean implements java.io.Serializable
{
    Private int Sno;
    Private String Sname;
    private String sadd;
    //write setXxx(-) and getXxx() methods
    ==========
}
```

Logic to transfer ResultSet object records to Arraylist object elements D.T.O class support

```
ResultSet rs=st.executeQuery("select * from student");
ArrayList al = new ArrayList();
while(rs.next())
 {
    StudentBean st=new StudentBean();
    //add each record values to D.T.O class obj
     st.setSno(rs.getInt(1));
     st.setSname(rs.getString(2));
     st.setSadd(rs.getString(3));
     //add D.T.O class obj to ArrayList
      al.add(st);
 }
```



Where did you use java bean in your project ?

(A) (i) As model layer resource in MVC-1, MVC-2 architecture based projects.

(ii) As D.T.O class or V.O class while transfering ResultSet object to Collection Frame work data structure as shown above.

(iii) As HB perisistance class in HB Programming.

(iv) As Spring Bean in spring Programming and etc...

---

* ~~java~~ can you tell me where exactly you have utilized the collection frame work data structures in your project?

(A) * java.util.Properties is required to maio while maintaing JDBC driver and database details outside the JDBC application.

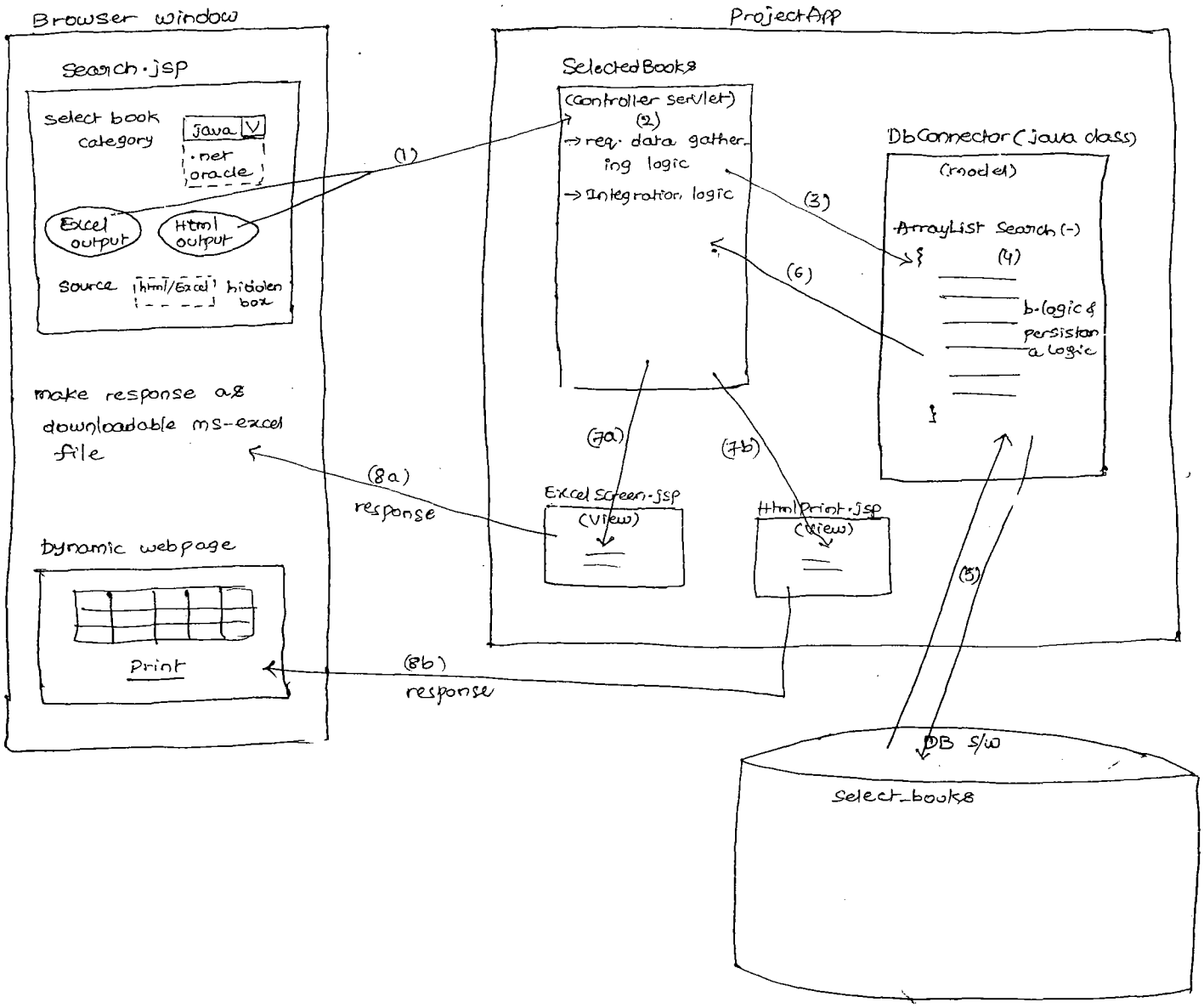
* ArrayList or Vector support will be taken while ~~tran~~ to send ResultSet object data over the network.

* To maintain JNDI properties, mail properties of Java mail api map data structures are required.

* To maintain application data (huge amount) with in the application Collection Frame work data structures are required.

* To maintain same form ~~data~~ page data across the multiple requests as a single value during the session Collection frame work data structures are required.

NOTE: To send collection framework datastructure over the network the objects that added to datastructures as elements must be serializable objects.



(1) Enduser selects the book categoery and submits the request to web application.

(2) The Controller Servlet traps and takes the request and reads form data from the form page.

(3) Controller servlet uses integration logic to call seach business method of DbConnector class having the selected book categeory as argument value.

(4), (5) The search business method uses business logic and persistance logic to gather book details from db table into ResultSet object and moves

that ResultSet object data to ArrayList by taking the support of D.T.O class.

(6) Search business method sends the ArrayList object back to the Controlle servlet.

(7a), (7b) The Controller Servlet sends ArrayList to result pages (ExcelScreen.jsp, HtmlPrint.jsp) as request attribute value.

(8a), (8b) Result pages sends response to browser window.

NOTE(1): ExcelScreen.jsp sends response to client as downloadable MS-Excel f where as HtmlPrint.jsp sends response to browser window as printable html table content.

The following observations can be done in the above project.

(i) web application is developed based on MVC2 architecture.

where

M → Model Layer → java class (DbConnector)

V → view Controller → jsp programs (3 prgs)

C → Controller Layer → servlet Prg (SelectedBooks)

(ii) Form Page is submitting request to web application through ordinary button taking the support of javascript.

(iii) Multiple buttons generated request related logics are differentiate in web application.

(iv) JavaScript also used for form validations (selecting item from in Selectbo is mandatory.

(v) D.T.O class support is taken to transfer ResultSet object data to Array.

(vi) Servlet chaining support is taken by Controller Servlet Communicating with result pages and request attributes are used to pass data from Con ller servlet to result pages.

(vii) Response is made as downloadable ms-Excel file and also made as th printable web page.

* For the source code of above project refer page nos 145-150.

* For other mini projects refer page nos 132, 133.

Custom Jsp Tag Library:

* Jsp tag library is a library that contains set of Jsp tags.
* Programmers can work with three types of tag libraries in their Jsp programs.
  (1) Built-in tag libraries of Jsp technologies.
      (All Standard Action tags of Jsp)
  (2) Third party supplied Jsp tag libraries
      Ex: Struts tag libraries, JSF tag libraries, JSTL and etc...
  (3) custom Jsp tag libraries
      (Developed by programmers)
* According to industry standard it is recommended to develop Jsp programs as Java codeless Jsp programs, to increase readability and easyness in Jsp program. That means we should not use scriptlet, declaration, expression tags in our Jsp programming.
* The built-in tags of Jsp are not at all sufficient to develop Jsp programs as Java codeless Jsp programs so we need to think above developing custom tag libraries or using third party tag libraries.
* In custom jsp tag libraries all jsp tags are custom Jsp tags.
* The Java class that defines the functionality of a Jsp tag is called as tag handler class. For this the class must extend from javax.servlet.jsp.tagext.TagSupport class.
* Every Jsp tag library contains one tld file (Tag Library Discriptor) having the configuration details of Jsp tag like Jsp tag name, attribute names, tag handler class name, list of possible values for attributes and etc...

Procedure to develop custom Jsp tag library

Step(1): Design the tag library and its Jsp tags.

<ABC> → should Print 1-10 numbers

<XYZ> → should print 10-1 numbers

Step(2): Develop tag handler classes defining the functionalities of Jsp tags in WEB-INF\classes folder

ABCTag.java

```
public class ABCTag extends TagSupport
{
   public int doStartTag()
    {
      ======  (g)
      ======
    }
    public int doEndTag()
    {
      ======  (g)
      ======
    }
}
```

XyzTag.java

```
public class XyzTag extends TagSupport
  {
     public int doStartTag()
       {
       }
     public int doEndTag()
       {
       }
  }
```

NOTE: The logic of doStartTag() executes when open JSP tag [<name>] encounters & in this method definition we can use method definitions and body of JSP tags to define the functionality of JSP tag. where as the doEndTag() exec when [</name>] encounters. In this method definition we can define the finishing functionalities

* doStartTag(), doEndTag() methods are container call back methods and life cycle methods of Tag handler classes.

Step(3): Configure all the JSP tags of custom JSP tag library in tld file.

WEB-INF\Sathya.tld (xml file) (Blue print)

<ABC> $\xrightarrow{(f)}$ ABCTag.class

<Xyz> → XyzTag.class

* Step(1) to step(3) talks about custom JSP tag library.

Step(4): Configure JSP tag library in web.xml file of web application.

```
<web-app>
    <taglib>
        <taglib-uri>demo</taglib-uri>   (d)
```

<taglib-location> /WEB-INF/sathya.tld </taglib-location>

(e) name & location of tld file

</taglib>

</web-app>

NOTE: Every JSP Taglibrary that is configured in web.xml file will be identified through its <taglib-uri> (like demo).

step (5): use the jsp tags of tag library in JSP programs of web application.

Test.jsp

<% @ taglib uri = "demo" prefix = "st" %> (c)

"demo" ↓ collect from web.xml

"st" → user defined

<st : ABC/> (b), (a)

<st : Xyz/>

→ (h) : output of <st:ABC> tag goes to browser window.

NOTE: prefix of the tag is useful to identify the tag library of JSP tag. when two jsp tags of two different jsp tag libraries contains same name and different functionalities in order to use both JSP tags in a single JSP program we can use prefixes to differentiate those tags.

* All custom JSP tags must be developed as action tags so we must use prefixes while working with those tags.

* In the above code the alphabets (a) to (h) indicates flow of execution related to <st:ABC> tag.

(a) → The <st:ABC/> tag encounters in the execution of JSP program.

(b) → The prefix st will be gathered from the tag.

(c) → Based on the prefix the <taglib-uri> demo will be gathered from <%@ taglib ... > tag (This tag is useful to specify the <taglib-uri> of tag library whose tags are going to be used in current JSP program).

(d), (e) → From web.xml file the tld file name and location will be gathered based on <taglib-uri>.

(f) → From tld file the tag handler class of <ABC> tag will be gathered. (ABCTag).

(g) → The code placed in Tag handler class executes through container callback methods.

(h) → The output generated by Taghandler class goes to browser window as the output of <st:ABC> tag.

* In every tag handler class the pageContext object is visible beca it is protected member variable of pre defined tag support class.
* Programmer uses this inherited pageContext object in Tag hand class to create/ to get access through other implicit objects of jsp li out, request, response and etc..
* For example application on custom jsp tag library development refer appl cation given in page nos 129-132.

JSTL

* The built-in jsp tags of jsp technology are not sufficient to make JSP programs as java code less jsp programs.
* Developing custom JSP tags and tag libraries is complex and time consuming process.
* To overcome above two limitations and to make JSP programs as java code le JSP programs use third party supplied (ready made) custom JSP tag libraries like JSTL, Display tags, struts tag Libraries and etc.

JSTL
- → Given by sun micro systems
- → is not part of JSP technology
- → should configure and should be used with JSP programs seperately.
- → Gives 4 no. of JSP tag libraries. They are
  - (1) Core JSP tag library
  - (2) SQL JSP tag library
  - (3) XML JSP tag library
  - (4) Formatting JSP tag library

* By using JSP tags of these four JSP tag libraries we can perform all opera tions using tags like variable declaration, write data to browser window, Cond tional operations, Iterational operations, Arithmetic and logical operations (also take the support of EL), DB interactions, xml processing, Formatting data and el
* Since JSTL gives readly available custom JSP tag libraries there is no need of designing JSP tags, developing tag handler classes and tld files.
* Core tag library is given for basic operations like variable declarations, flow of control, conditions and iterations.
* sql tag library is given for DB interactions.
* xml tag library is given for xml processing.

Tag Library	tld file name	recommended prefix	jar files
core	c.tld	c	jstl.jar, standard.jar, saxpath.jar, jaxen-full.jar
SQL	sql.tld	sql	"
xml	x.tld	x	"
fmt	fmt.tld	fmt	"

* Fmt tag library is given for specifying Date formats, Time formats, currency symbols, NumberFormats and etc...
* For tags of various JSTL tag libraries refer page nos 116 to 119.
* we can gather the above given four jar files JSTL.jar, standard.jar, saxpath.jar, jaxen-full.jar from java.sun.com or oracle.com website.

Procedure to work with JSTL tags in java web application

(1) Gather the above said four jar files keep them in WEB-INF\lib folder.
(2) Extract the tld file from standard.jar file and place in webapplication like WEB-INF folder.
(3) Read the documentation about JSTL tags.
(4) Configure JSTL supplied tag libraries in web.xml file having taglib uris.
(5) use JSTL tags the jsp programs of web application.

* while working with JSTL we can use more implicit object along with the regular 9 implicit objects. They are
(1) requestScope
(2) param
(3) paramValues
(4) header
(5) headerValues
(6) Cookie
(7) sessionScope and etc...

* Tags are not there in tags can not be given to perform arithmetic operations in our JSP programs. For this purpose we can take the support of EL which can be used with tags or without tags.

Syntax

$ { ------ < EL Code > ----- }

* For example application on JSTL tags refer ~~to~~ examples given in 120-li of Booklet.
* The implicit object param can be used to gather single request parameter value. where as the implicit object paramValues can be used to gather all request parameter values.

Web Sphere server

Type: Application server software
version: 6.x (Compatable with jdk 1.4)
Vendor: IBM
default port no: 9080 for web applications
Commercial software
jar file that represents jee apis : j2ee.jar
download s/w as ZIP file from www.IBM.com website.
Allows to create domains.
If websphere s/w is installed as windows service then each domain mus be started services of Control panel.

* To install websphere 6.0 use install.exe file of WAS folder that comes by extracting IBM Websphere Application Server V6.0 win.rar/ zip file.
* To create domain in web sphere server

Start → programs → IBM web Sphere → Application Server V6 → Profile creation wizard ....

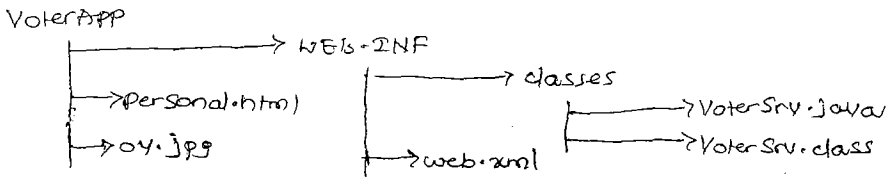
* In web sphere server war file based console deployment is possible.

procedure to deploy java web application in websphere 6.0

Step(1): set path to jdk 1.4 that comes with websphere s/w installation.

> PATH = D:\Program Files\IBM\AppServer\java\bin

Step(2): Prepare war file on deployment directory structure of web application.



NOTE: Compile VoterSrv.java in jdk1.4 environment.

<!DOCTYPE ... ...> statement in web.xml file is mandatory for websphere server.

> jar cf VoterApp.war .

step(3): start the domain server of websphere from services of Control panel.

Control panel → performance and maintainance → Administrative tools → Services → IBM Websphere server → start.

step(4): open administrative console of domain Server.

start → All programs → IBM websphere → application server V6 → profiles → mydomain → Administrative console → UserID: xyz

(mydomain: user defined)

step(5): Deploy the web application

Admin console → Install new application → Browse and select VoterApp.war file → Context root: VoterAPP → next → next → Continue → next → next → next → Finish → Save to master Configuration → save.

step(6): keep the deployed VoterApp application in running mode.

Admin console → Enterprise applications → ☑ VoterApp.war → start

step(7): Test the web application.

open browser window → http://localhost:9080/VoterApp/person.html

what is the difference between encodeURL(-), encodeRedirectURL(-) methods of response object ?

(A) encodeURL(-) is useful to append session id to the given URL. This is useful while working with HttpSession with URL rewriting technique.

encodeRedirectURL(-) is useful to send session id from one web application to another application then they are interact with each other with help of sendRedirect(-). This method is useful to make session obj of one web application to another webapplication.

Example on encodeRedirectURL(-) (In SrvI prg of WA1 web application).

```
String enurl = res.encodeRedirectURL("http://machine20:7001/WA2/srv2");
res.sendRedirect(enurl);
```

The above code sends session id from WA1 web app to WA2 web app.

Annotations

* Data about data is called as meta data.
* Gathering more details about already given details is called meta data operation.
* In JDBC level we can perform meta data operations in 3 modes.
  (1) Database meta data
  (2) Parameter meta data
  (3) ResultSet meta data
* Configuration of resource programs in xml files comes under meta data operations.
  Ex: By configuring servlet, JSP programs in web.xml we are passing more details about those programs to underlaying container or server software. This is nothing but meta data operations.
* While working with advanced technologies we generally use xml files to perform resources configuration based meta data operations.
* Annotations are the java statements and they are alternate for the xml file based meta data, resources configuration operations.

Syntax:

@ <annotation name> (param1 = val1 , param2 = val2, - - - - -)

* To read and process xml documents we need xml parser which is a heavy weight software so it may kill the performance of the application but xml files give good flexibility of modification without disturbing the java source code.
* Annotations will be recognized by java tools or underlaying containers automatically so they give good performance but they give bad flexibility of modification.

Two types of annotations

(1) Documentation related Annotations (should be used in /** - - - - - */ Comment)
(2) Programming related Annotations (can be used in java source code level)

(1) → ex: @author, @param, @see, @since, @return . . .

(2) → ex: @ override, @ webServlet, @ ServletFilter, @ InitParam and etc..

* Annotations for documentation are given from JDK 1.1 onwards (rt.jar file of JDK s/w).
* Annotations for programming are given from jdk 1.5 so all technologies that are compatable with jdk 1.5 gives support for annotations.
Ex: EJB 3.x, Struts 2.x, spring 2.5, 3.x, GJB 3.3+, servlet 2.5, servlet 3.x and etc
* Each annotation is a special java interface (with @interface keyword) and parameters of annotation will be declared as java methods.
* Parameters of annotations are like attributes of xml tags that means they can be used to configure additional details about resource.
* Recognizing annotations and recognizing annotation related resources and applying functionality on them is the responsibility of underlaying jdk or jre or container software.
* we can apply annotations at 3 levels.
(1) Resource level (class level / Interface level)
(2) Method level
(3) Field level (Member variable level)
* Annotations helps the programmer to configure the resources as required for underlaying container software or framework software.

Setup required to servlet 3.0 based servlet programs (no web.xml file)

JDK 1.6 + (jdk 1.6)

Tomcat 7.0 + (The server that is compatable with servlet 3.x specification).

GlassFish 3.x

Important annotations of servlet 3.x programming

@ WebServlet → to configure servlet program with logical name and url pattern.

@ InitParam → to configure init parameters of servlet program.

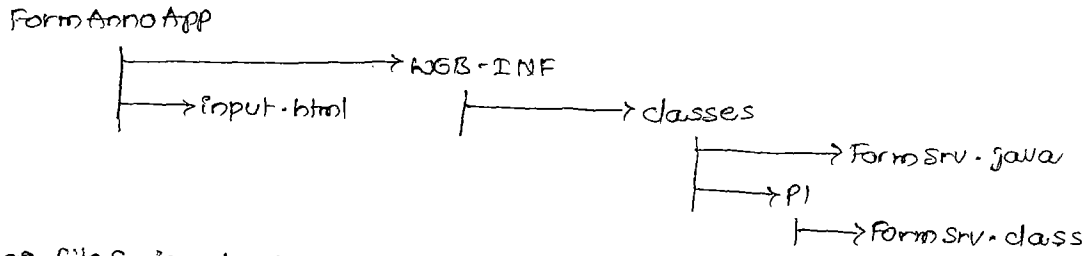
@ ServletFilter → to configure servlet filter program.

@ WebServletContextListener → to configure ServletContextListeners.

* All these annotations are part of servlet 3.x api and can be collected from <tomcat_home (7)>\lib\ servlet-api.jar.

* The main feature of servlet 3.x is good support for annotations and ability to develop servlet programs based web application without web.xml file.
* From jdk 1.5 onwards each java package contains classes, interfaces, annotations and enums.
* The servlet api 3.0 is given having 3 packages.
  javax.servlet, javax.servlet.http and javax.servlet.annotation.
* To get servlet 3.x api documentation refer javaees api docs...

Example application (using servlet 3.x annotations)



jar files in class path

<Tomcat 7_home>\lib\servlet-api.jar file.

Deploy above application in Tomcat 7.0 or Glassfish 3.x server.

input.html

```
<form action="test1">
    Name : <input type="text" name="username"> <br>
    <input type = "submit" value="check">
</form>
```

FormSrv.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet
@WebServlet(name="abc", urlPatterns={"/test"})
public class FormSrv extends HttpServlet
  {
    public void doGet(-,-) throws SG, IOE
      {
        String username = request.getParameter("user.name");
        response.setContentType("text/html");
```

```
        PrintWriter pw = response.getWriter();
        pw.println("Hello Mr:" +username+ " Today's Date :: " +new java.util.Date());
        pw.close();
    }
    public void doPost(--) throws SG,IOE
    {
        doGet(--);
    }
}
```

Request URL

http://localhost:2525/FormAnnoApp/input.html

* For related info on servlet 3.0 annotation refer supplementary hand out given on 18-12-2011.
* NetBeans 6.9 + is giving support for annotations based servlet Programming.
* Reflection means mirror image.
* Reflection api application is capable of gathering given java class or interface details programatically and dynamically.
* Being from one application if you want to gather certain java class details like member variables, methods and etc.. details programatically then develop the code of that application by using reflection api.
* To develop and use our own annotations we need to develop one simulator class called container on the top of JRE then only we can provide functionality to that annotation.

Procedure to develop user defined annotation having the following functionality

@ sum (num1 = 10, num2 = 20)
int result → should assign 30 to result variable.

@ sum (num1 = 30, num2 = 40)
int res → should assign 60 to res variable.

Step(1): make sure that jdk 1.5 + software is available.

Step(2): Design your annotation.

```
package P3;
import java.lang.annotation.*;
@Retention(RetentionPolicy.RUNTIME)
```

makes annotation as runtime annotation

```
@Target(ElementType.FIELD)  -> this annotation is applicable only on fields.
public @interface Sum {
    int num1() default 0;
    int num2() default 0;
}                                   } parameters.
```

Step(3): Develop your own Container class.

```
package P3;
import java.lang.reflect.Field;
public class Container
{
  public static void addThis(Object theObject)
  {
    try{
      //get access to all the member variables of current obj (Test class obj)
      Field[] fields = Class.forName(theObject.getClass().getName()).getDeclaredFields();
      for(int i=0; i<fields.length; i++)
      {
        //get access to @Sum annotation
        Sum obj = fields[i].getAnnotation(Sum.class);
        if(obj!=null)
        {
          //getting values from annotations parameter
          int val1 = obj.num1();
          int val2 = obj.num2(); int tot = val1+val2;
          //setting it to our class variable (like result)
          fields[i].setInt(theObject, tot);
        }//if
      }//for
    }catch(Exception e){e.printStackTrace();}
  }
}
```

Step(4): use the annotation in your application

```
package P3;
public class Test
{
  @Sum(num1=15, num2=10)
  static int result;
  public static void main(String args[])
  {
    Test t1 = new Test();
    Container.addThis(t1); //activate user-defined Container class
    S.o.p("The sum is:" +result);
  }
}
```

Step (5): Compile and execute the application.

```
>javac -d. *.java    (Compilation)
>java P3.Test        (execution)
```


—— • ——

* There are multiple ways to know the details of java class or interfac

They are

(1) By working with javap tool

(2) By working with java api documentation.

(3) By opening source code of java resources.

javap :

* It is a built-in tool of jdk software.

Ex: javap java.lang.String

javap java.io.Serializable

javap MyTest

---

* In jdk software installation we can use java_home\src.zip file to get the source code of all the built-in classes and interfaces of java ap

* When all the above three techniques are used to gather details about certain java class or interface we can not use those details as input values in our java application.

* Reflection means mirror image.

* Reflection api based java applications can gather the internal details abo given java class or interface programatically and dynamically. and these deta can be used in the application as input values for other requirements.

* Each reflection api based java application acts as mirror image having the ability to gather details about given class or interface dynamically.

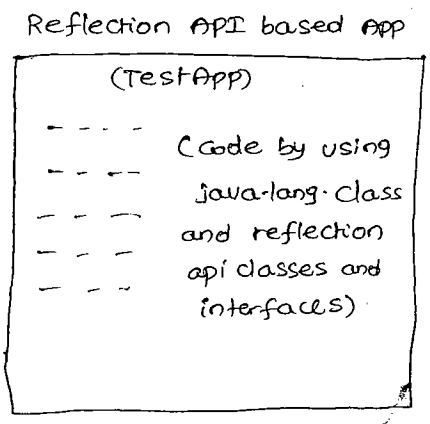
* Reflection api means working with classes and interfaces of java.lang. reflect package.

* java.lang is default package in java applications but its sub packages ar not default packages.

```
public final class Test
 {
   float static int a;
    private float b;
     Test()
      {
      }
```

```
    Test (int x)
    {
    ==
    }
    public int sum (int x, int y)
    {
      ==
      ==
    }
}
```

* In java environment member variables are technically called as fields and constructors are called as creation methods.



>java TestApp.java

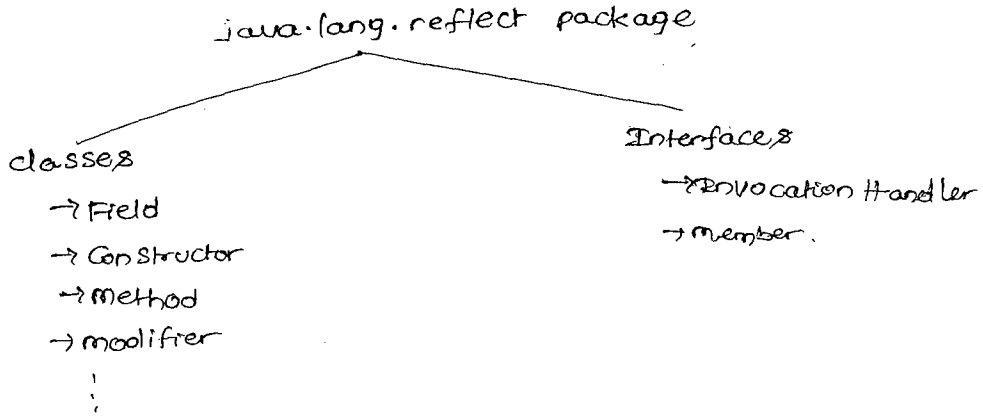
>java TestApp java.lang.System → Command line argument value

→ prints all the details of java.lang.System class as output.

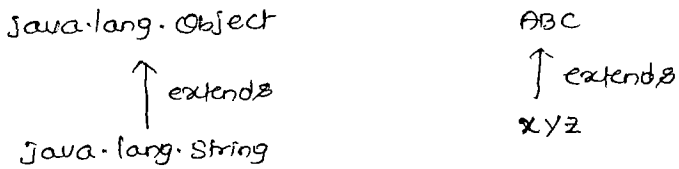
>java TestApp java.lang.Runnable

→ prints all the details of java.lang.Runnable (I) as output.

* Reflection api is popularly used in the development of java based software products like debuggers, classBrowsers, GUI Builders and etc....
* Debugger allows us to trace flow of execution in the application.
* classBrowser is nothing but hint box that comes in IDE s/w.
* GUI Builder allows the programmer to design the application through drag and drop operations.
* Every IDE contains Debugger, classBrowser, GUI Builder products. This indicates the IDE s/w internally using reflection api support.
* we can gather the basic details about given class or interface by using various methods of java.lang.class but use classes and interfaces of java.lang.reflection api to gather advanced details.



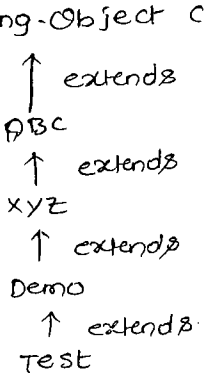
write a java application to gather super class name of given java class




```
// Appl.java
  public class Appl
  {
     public static void main (String args[]) throws Exception
     {
        // load given java class into App
         Class c = Class.forName (args[0]);
         // get super class of given class
         Class sc = c.getSuperclass();
         // print given class, super class name
         system.out.println ("given class name:" + c.getName());
         System.out.println ("super class name:" + sc.getName());
     }//main
  }// class
  // > javac Appl.java
  // > java Appl java.lang.String  → command line argument value.
  // > java Appl java.lang.System
```

* when the object of java.lang.Class points to class or interface and to get that class name or interface name as string value use getName() method call on java.lang.Class object.

write a java application to get all the classes of inheritance hieardy for a given java class




```
//APP2.java
public class APP2
{
   public static void main(String args[]) throws Exception
   {
      //load given java class into APP
        class c = class.forName(args[0])
        //get super class
        class sc = c.getSuperclass();
        // print all the classes of inheritance hierarchy
         system.out.println("classes in the inheritance hierarchy for" + args[0] +
                                                    " class are:");
         while(sc!=null)
          {
            system.out.println("\t\t" + sc.getName);
             c=sc;
             sc=c.getSuperclass();
          }//while
   }//main
}//class.
```

```
//>javac APP2.java
//>java APP2 java.lang.Integer
//>java APP2 java.applet.Applet
```

* Object of java.lang.class can represent a concrete class or interface or abstract class or data type.

write a java application to get list of java interfaces implemented by the java given java class

```
public class Test implements X, Y, Z
  {
    ==
  }
```

```
//App3.java
public class App3
 {
   public static void main(String args[]) throws Exception
     {
       //load the given java class into App
        class c = class.forName(args[0]);
        //call getInterfaces() of java.lang.class
        class inter[] = c.getInterfaces();
        //print interfaces names
         for(int i=0; i< inter.length; i++)
          {
            system.out.println("/t\t" + inter[i].getName());
          }//for
     }//main
 }//class
//> javac App3.java
//> java App3 java.lang.String
//> java App3 java.lang.Thread
```

* int[] means all elements in that array are int values.
* java.lang.class object[] means all elements in that array are the objects of java.lang.class.
* public, final, abstract are the modifiers of java class. But we can not apply final and abstract modifiers at a time on a java class because final class does not support inheritance and abstract class needs the suppo of inheritance (we can not write subclass to final class but we need to write subclass to abstract class to implement abstract methods).

write a java application to get the modifiers of given java class

```
//App4.java, import java.lang.reflect.*;
public class App4
 {
   public static void main(String args[]) throws Exception.
     {
       //load the given java class into App
        class c = class.forName(args[0]);
        //call getModifiers() of java.lang.class.
         int x = c.getModifiers();
```

→ returns int value representing the modifiers of given java class.

```
            //print modifiers
             if (Modifier.isPublic(x))
                 System.out.println("\t public");
             if (Modifier.isAbstract(x))
                  System.out.println("\t abstract");
             if (Modifier.isFinal(x))
                  System.out.println("\t final");
         }//main
     }//class
//>javac App4.java
//>java App4  java.lang.String.
//>java App4  java.lang.Thread.
```

* isXxx(-) methods of Modifier class are boolean methods. Checking whether the given numeric value represents certain modifier or not.
* By default all interfaces are abstract.
* In java there is no terminology called access specifier.
* public, private, protected, final, static, synchronized, volatile, transient and etc.. keywords are called modifiers in java.
* The object of java.lang.reflect.Field class can represent one member variable of given java class.

write a java application to get the details of member variables / fields available in java class

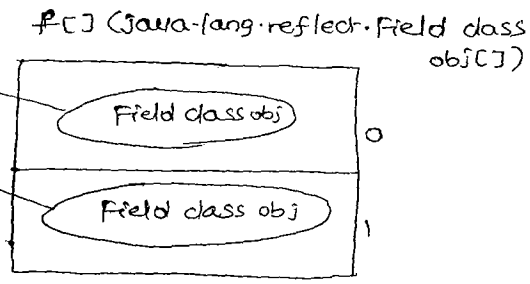
```
//App5.java
 public class App5
 {
    public static void main(String args[]) throws Exception.
     {
        //load the given java class into App
         Class c = Class.forName(args[0]);
         //call getDeclareFields() of java.lang.class
         Field f[] = c.getDeclareFields();
         //Print Field details
         for(int i=0; i< f.length; ++i)
          {
             int x = f[i].getModifiers(); //gives modifiers of each field.
             String type = f[i].getType().getName(); //gives datatype of each field.
```

```
String name = f[i].getName(); // gives name of each field
if (Modifier.isPublic(x))
    System.out.println("public");
- - - - -
- - - - -
System.out.println(type+"  "+name);
} // for
} // main
} // class
```

```
// > javac App5.java
// > java App5 java.lang.String
```

Ex:

```
public class Test
{
  private int a;
  static final float b;
  ------ // methods
}
```



Write a java application to gather all the method details of given java cla

```
//App6.java
import java.lang.reflect.*;
public class App6
{
  public static void main(String args[]) throws Exception
  {
    // load the given java class into App
    class c = Class.forName(args[0]);
    Method m[] = c.getDeclareMethods();
    // print method details
    for(int i=0; i<m.length; ++i)
    {
      int x = m[i].getModifiers() // gives modifiers of each method
      String rtype = m[i].getReturnType.getName(); // gives return type
      String mname = m[i].getName(); // gives method name.
      class p[] = m[i].getParameterTypes(); // gives parameter types.
      if(Modifier.isPublic(x))
        S.o.p("public");
```

```
- - - -
- - - -
if (modifier.isFinal(x))
    s.o.p("final");
    - - - -
    - - - -
s.o.p(rtype + "  " + mname + "(");
for(int k=0; k<p.length; ++k)
   {  s.o.p(p[k].getName() + " ");
   }// for
   System.out.println(")");
}// for
}// main
}// class.
```

```
// > javac APP6.java
// > java APP6 java.lang.String
// > java APP6 java.lang.Runnable
```

Memory diagram

```
public class Test
{
  public void hello()
   {
   }
  public static final int Cin sum(int x, int y)
   {
     return x+y;
   }
}
```