# JSP

## (Java Server Pages)

**JSP is a tag based web technologies to develop web application for providing dynamic response.**

**Compare to Servlet, JSP is used to develop web applications within the less time.**

**The reason is if we go for Servlet we need to import statement, class declarations, method declarations, exception class's declarations and finally writing logic in the methods. In JSP, we are always writing logic directly within the tags.**

**JSP is not a replaced technology of Servlet.**

**It is not extension technology of a Servlet.**

**It is just alternative (one more way of developing web application) technology.**

**In Servlet we are using mainly two types of coding.**

**a. processing code ( java code)**

**b. printing code( html code)**

**Whatever the request ( input data ),which is coming from browser, that request is processing and generating response for this purpose Servlet' s uses java code for handover the response to end user (browser) Servlet required html code.**

**In JSP we used three types of tags.**

1. **html tags**
2. **JSP tags.**
3. **java code**

**Html tags are useful for displaying the information on browser.**

**JSP tags and java code are useful for processing the request and generating response.**

**Syntax of JSP tags:**

```
<%        %>
```

**No need mention any information about JSP in web.xml file, the reason is we are communicating with JSP by using JSP file name.**

**Example:**

```
<html>
    <h1>Result is: 50+60</h1>
      Result is<% = 50+60 %>
</html>
```

**If we are writing any code outside of JSP tags that code will treated as html information.**

**Result is: <% = 5+60%>**

**In the above Result is: will treat as html code.**

**To communicating with JSP we don't required any special configurations in web.xml file. No need to create any WEB-INF or any other folders.**

**We can communicate JSP with the help of JSP file name.**

**JSP file always having extension like ".jsp". It is mandatory.**

**<u>Advantages:</u>**

1. **Faster development**
2. **Auto Compile**
3. **Auto reload**

**The above features are providing by the all servers.**

**<u>Steps for developing JSP:</u>**

1. **Developing dynamic web project (TestJsp) and extract the project hierarchy.**
2. **Right click on WebContent**
3. **Select new**
4. **Go to other**
5. **Select File option under general option**
6. **Click on next**

**7. Give file name as Test.jsp**

**8. Click finish**

**9. Write some code in Test.jsp**

```
<html>
<body bgcolor='yellow'>
<form>
<h1><MARQUEE>WELCOME TO JSP PROGRAMMING</MARQUEE></h1>
</form>
</body>
</html>
```

**10. Save**

**11. Start the server**

**12. Under browser type bellow url**

**http://localhost:1111/TestJSP/Test.jsp**

**13. Check output on browser it print**

**WELCOME TO JSP PROGRAMMING**

**14. Go to Test.jsp and rewrite the code as bellow**

```
<html>
<body bgcolor='yellow'>
<form>
```

**<h1><MARQUEE>WELCOME TO JSP PROGRAMMING</MARQUEE></h1>**

**<h1><MARQUEE>AUTOLOADING AND FAST DEVEOPMENT</MARQUEE></h1>**

**<h1><MARQUEE>AUTOCOMPILE</MARQUEE></h1>**

**</form>**

**</body>**

**</html>**

**15. Come to browser to just refresh it.**

**16. It will out put like**

**WELCOME TO JSP PROGRAMMING**

**AUTOLOADING AND FAST DEVEOPMENT**

**AUTOCOMPILE**

**That means the above step will prove that auto compile and reload.**

**INTERNAL PROCESS OF SERVER (JSPCONTAINER & SERVLET CONTAINER):**

**At end of the JSP file it converted into servlet.**

**JSP container duties:**

**1. Verification:**

**The code which we written in the .jsp file is valid or not.**

**2. Translation:**

**“.jsp” will be converted into .java file.**

**Ex: Test.jsp (JSP file name)**

**Test_jsp.java(.java file name)**

**3. Compilation:**

**Test_JSP.java will converts into .class file like bellow**

**Test_jsp.java (.java file name)**

**Test_jsp.class (.class file name)**

**4. Configuration:**

**Handover Test_JSP.class file to Servlet Container.**

**5. Loading:**

**This will doing by the Servlet container it will load the .class bytecode.**

**6. Instantiation:**

**Creting object for Test_JSP**

**7. Initialization:**

**Filled with initialization values**

**8. Execution:**

**Test_JSP will be executing**

**9. Destroy:**

**The services of Test_JSP will be closed**

**JSP life cycle methods:**

**public void jspInit()**

**public void _jspService(HttpServletRequest req, HttpServletResponse res)throws ServletException, IOException;**

**public void destroy()**

**jspInit() method is used for providing JSP implemented class object initialization logic.**

**_jspService() is used for providing the request processing logic for executing the JSP page request.**

**jspDestroy method is used to JSP implemented class for releasing logic.**

**If we want to see PIC go and check in bellow folder:**

**D:\NARESH-I-TECHNOLGY\CORE-JAVA\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina\localhost\TestJSP\org\apache\JSP**

**The above methods are not called by servlet container directly, these are called from servlet life cycle methods.**

**to called these methods every server vendor will provide implementation class and override the servlet life cycle methods.**

**init(ServletConfig) --> calls ---> JspInit()**

**service(HSReq,HSRes)---> calls-->_JSPService()**

**destroy() ---> JSPDestroy()**

**_JSPService(-,-) throwing ServletException only, the reason is it called from service(-,-).**

**we can able to write JspInit() and JSPDestroy() methods by using this tag**

**<%! ...%>**

**but we can not override _JSPService(). the reason is server provide JIC always having _JSPService()**

**JSP implementation class structure:**

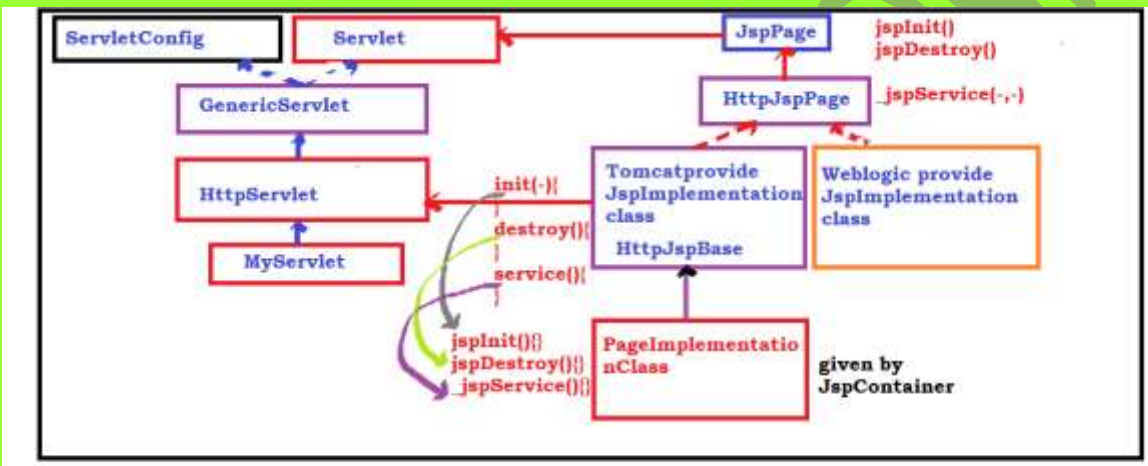
**<html>**

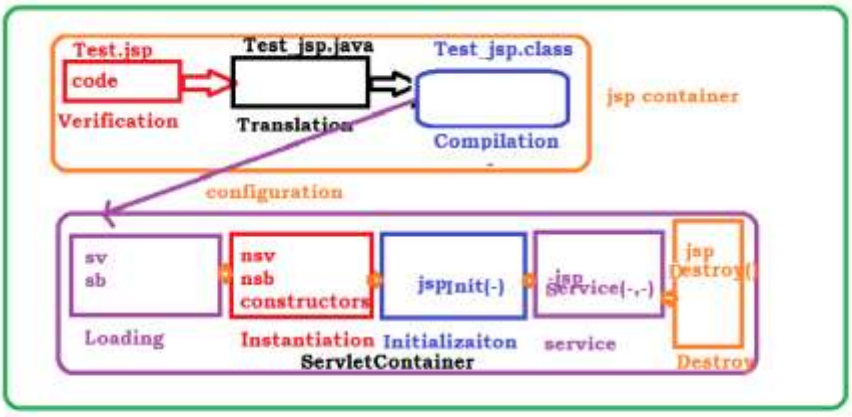
**<body>**

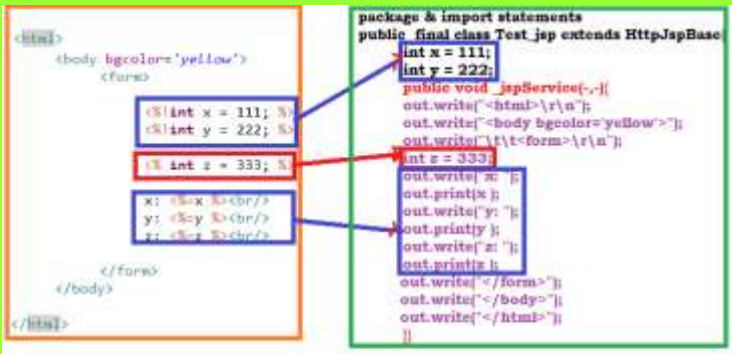
**<%! int x = 10; %>**

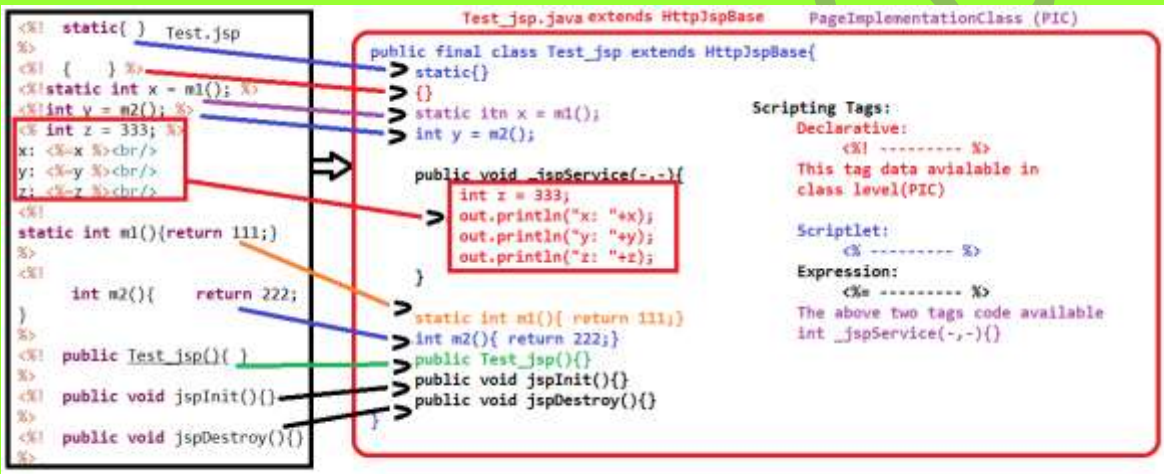
```
<% int y = 20; %>

x: <%= x %> <br/>

y: <%= y %>

</body>

</html>
```

---






**Q) Can we override _jspService(-,-) in our .jsp file?**

**NO.**

**The reason is by default JspContainer will place _jspService(-,-) in our PageImplementationClass, again programatically if we write _jspService(-,-) we will get error like duplicate methods.**

**But we can write jspInit(-) & jspDestroy().**
**The reason by default JspContainer not place the above two methods into our PageImplementationClass.**

**<html>**

```
<body>
	<%! static{
		System.out.println("JSP page is loaded-static block");
	}
	%>
	<%! public test_JSP(){
		System.out.println("JSP page instantiated: constructor");
	}
	%>
	<%! public void JspInit(){
		System.out.println("JSP page initialized");
	}
	%>
	<%!
	{
	System.out.println("JSP page initialized: non-static block");
	}
	%>
```

```
<%!
public void JSPDestroy(){
        System.out.println("JSP page destroyed:
destroy");
}
%>
<%="request processed: service"

%>
</body>
</html>
```

**1st request: JSP page is loaded-static block**

**JSP page initialized: non-static block**

**JSP page instantiated: constructor**

**JSP page initialized**

**request processed: service**

**2nd request:**

**request processed: service**

**3rd request:**

**JSP page destroyed: destroy**

**JSP page is loaded-static block**

**JSP page initialized: non-static block**

**JSP page instantiated: constructor**

**JSP page initialized**

*******request processed: service**

**How to configure JSP file in to web.xml file and what is the use:**

**We can configure .jsp file within the web.xml file by using <jsp-file>tag.**

**<jsp-file> is the sub tag of <servlet>**

**Writing .jsp file name in the browser URL and making a request server and getting response from server will take some time, to avoiding this time consuming problem and meanwhile of deploying our .jsp project into server if we want to do the following phases and making my .jsp for ready to provide response to end user we need to configure .jsp file in web.xml file.**

**Once we configure .jsp file information in web.xml file the following will be happen by both jsp container and servlet container.**

**Verification, Translation, compilation, configuration, loading, instantiation, initialization.**

**ex:**

**<servlet>**

**<servlet-name>test</servlet-name>**

**<jsp-file>/Test2.jsp</jsp-file>**

**</servlet>**

**JSP file name like Test2.jsp in <jsp-file> tag must be starts with '/'.**

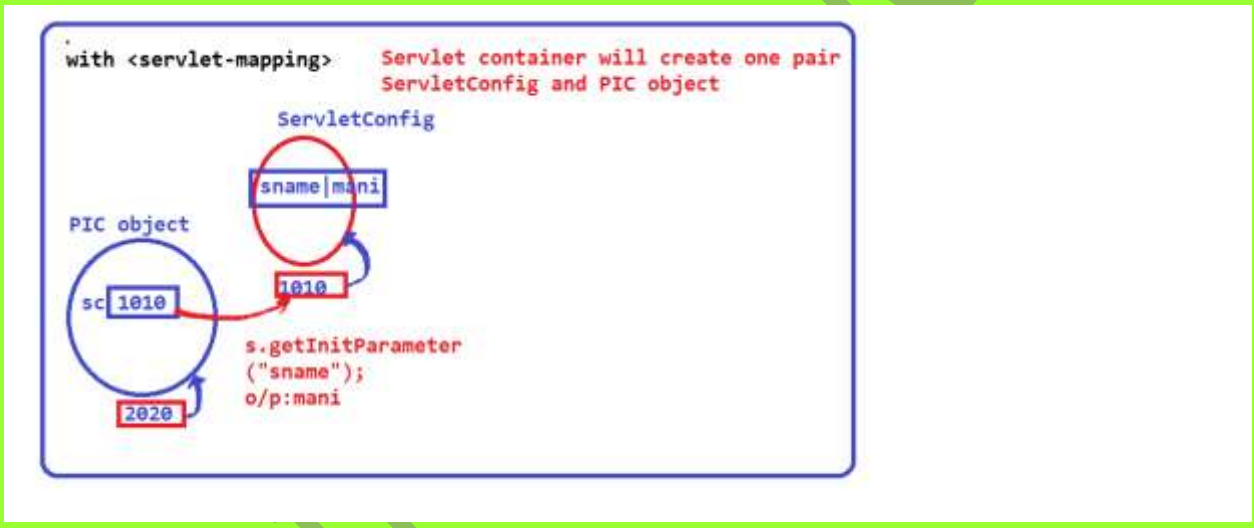
---

**Test.jsp:**

```
<html>
<body bgcolor='yellow'>
      <%! public void init(){
                  System.out.println("this init method");
                  ServletConfig sc = getServletConfig();
                  System.out.println("servletconfig: "+sc.hashCode());
                  System.out.println("currentobject: "+this.hashCode());
                  String studentName = sc.getInitParameter("sname");
                  System.out.println("StudentName: "+studentName);

      } %>
      <%
            System.out.println("this _jspService method");
            ServletConfig sc = getServletConfig();
            System.out.println("servletconfig: "+sc.hashCode());
            System.out.println("currentobject: "+this.hashCode());
            String studentName = sc.getInitParameter("sname");
            System.out.println("StudentName: "+studentName);

      %>
</body>
</html>
```

## Web.xml:

```
<web-app>
      <servlet>
            <servlet-name>ts</servlet-name>
            <jsp-file>/Test.jsp</jsp-file>
            <init-param>
                  <param-name>sname</param-name>
                  <param-value>mani</param-value>
            </init-param>
      </servlet>
      <servlet-mapping>
            <servlet-name>ts</servlet-name>
            <url-pattern>/Test.jsp</url-pattern>
      </servlet-mapping>
</web-app>
```




```
INFO: Initializing ProtocolHandler ["ajp-nio-2015
Oct 25, 2017 11:59:50 AM org.apache.tomcat.util.n
INFO: Using a shared selector for servlet write/r
Oct 25, 2017 11:59:50 AM org.apache.catalina.star
INFO: Initialization processed in 866 ms
Oct 25, 2017 11:59:50 AM org.apache.catalina.core
INFO: Starting service [Catalina]
Oct 25, 2017 11:59:50 AM org.apache.catalina.core
INFO: Starting Servlet Engine: Apache Tomcat/8.5.
this init method
servletconfig: 24987666
currentobject: 911592
StudentName: mani
Oct 25, 2017 11:59:52 AM org.apache.coyote.Abstra
INFO: Starting ProtocolHandler ["http-nio-2016"]
Oct 25, 2017 11:59:52 AM org.apache.coyote.Abstra
INFO: Starting ProtocolHandler ["ajp-nio-2015"]
Oct 25, 2017 11:59:52 AM org.apache.catalina.star
INFO: Server startup in 1483 ms
this _jspService method
servletconfig: 24987666
currentobject: 911592
StudentName: mani
```

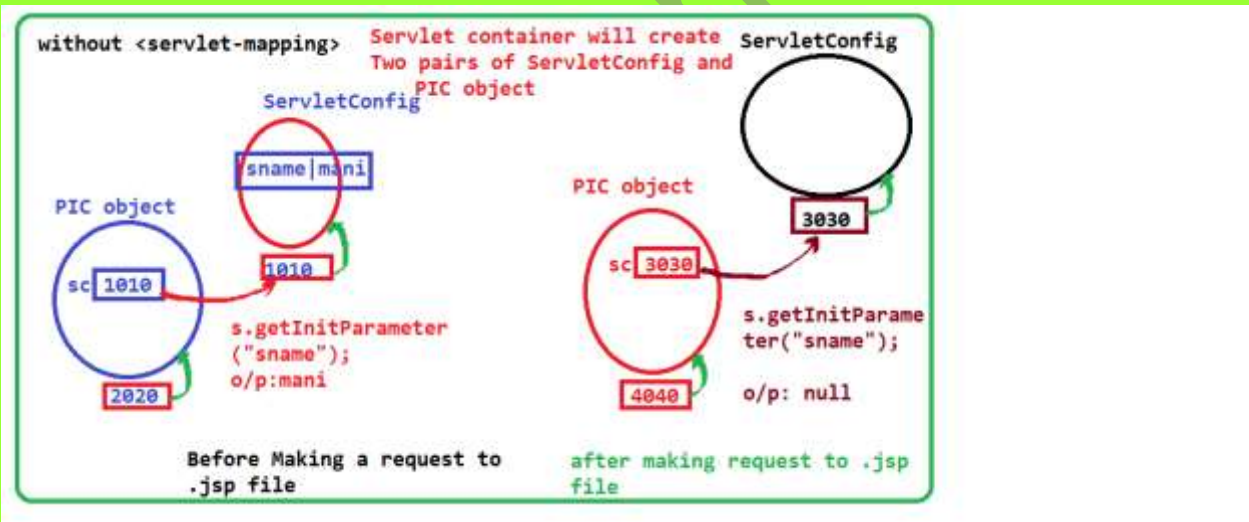
## Without <servlet-mapping>

## Test.jsp:

### Same as above program

## Web.xml:

```
<web-app>
      <servlet>
            <servlet-name>ts</servlet-name>
            <jsp-file>/Test.jsp</jsp-file>
            <init-param>
                  <param-name>sname</param-name>
                  <param-value>mani</param-value>
            </init-param>
      </servlet>
<web-app>
```




**program:**

**Test2.jsp same as bellow program**

**web.xml:**

**----------**

```
<web-app>
    <servlet>
        <servlet-name>test</servlet-name>
        <jsp-file>/Test2.jsp</JSP-file>
    </servlet>
    <servlet-mapping>
        <servlet-name>test</servlet-name>
        <url-pattern>/Test2.jsp</url-pattern>
    </servlet-mapping>
</web-app>
```

**If we load .jsp file before request coming to server about .jsp file, we need to configure .jsp file in web.xml file.**

**It loading .jsp we required only <servlet> is enough.**

**No need of <servlet-mapping> tag.**

**If we are configuring JSP file with help of <servlet-mapping>, then container will create**

**only one pair of JSP object(JSP Implementation class object--servlet),**

**ServletConfig objects are created.**

**If we have any initalization values those are playced into ServletConfig object**

**in the form of key-value pair of combination.**

**the following diagram shows how servletconfi objects holds the initialization information.**

```
<html>
<body>
    <%! static{
        System.out.println("static block");
        }
```

```
%>
<%! public Test2_JSP(){
System.out.println("constructor");
}
%>
<%! {
System.out.println("non-static block");
} %>
<%! public void JspInit(){
        System.out.println("JspInit()");
ServletConfig cfg = getServletConfig();
System.out.println("config: "+cfg);
System.out.println("this: "+this);
System.out.println("ename:
"+cfg.getInitParameter("ename"));
} %>
<%! public void JSPDestroy(){
        System.out.println("JSPDestroy");
} %>
<%
System.out.println("_JSPService()");
```

```
        ServletConfig cfg = getServletConfig();
        System.out.println("config: "+cfg);
              System.out.println("this: "+this);
        System.out.println("ename:
"+cfg.getInitParameter("ename"));


         %>
         <%= "program executing successfully:changed" %>
</body>
</html>
```

```
static block
non-static block
constructor
JspInit()
config:
org.apache.catalina.core.StandardWrapperFacade@8c0b
5a
this: org.apache.jsp.Test2_JSP@3d60c4
ename: ramchandra
```

**_JSPService()**

**config: org.apache.catalina.core.StandardWrapperFacade@8c0b5a**

**this: org.apache.jsp.Test2_JSP@3d60c4**

**ename: ramchandra**

**_JSPService()**

**config: org.apache.catalina.core.StandardWrapperFacade@8c0b5a**

**this: org.apache.jsp.Test2_JSP@3d60c4**

**ename: ramchandra**

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

**if we are not configure .jsp file with the help of <servlet-mapping>,**

**for each and every request container will create different servletobject and servletconfig objects.**

**<html>**

**<body>**

**<%! static{**

**System.out.println("static block");**

```
        }
    %>
    <%! public Test2_JSP(){
    System.out.println("constructor");
    }
    %>
    <%! {
    System.out.println("non-static block");
    } %>
    <%! public void JspInit(){
        System.out.println("JspInit()");
    ServletConfig cfg = getServletConfig();
    System.out.println("config: "+cfg);
    System.out.println("this: "+this);
System.out.println("ename:
    "+cfg.getInitParameter("ename"));
    } %>
    <%! public void JSPDestroy(){
        System.out.println("JSPDestroy");
    } %>
    <%
```

```
        System.out.println("_JSPService()");
       ServletConfig cfg = getServletConfig();
       System.out.println("config: "+cfg);
             System.out.println("this: "+this);
       System.out.println("ename:
"+cfg.getInitParameter("ename"));


        %>
 <%= "program executing successfully:changed" %>
</body>
</html>
static block
non-static block
constructor
JspInit()
config:
org.apache.catalina.core.StandardWrapperFacade@19be
b29
this: org.apache.jsp.Test2_JSP@16c01c8
ename: ramchandra
```

**static block**

**non-static block**

**constructor**

**JspInit()**

**config: org.apache.catalina.core.StandardWrapperFacade@8cb492**

**this: org.apache.jsp.Test2_JSP@1a2edef**

**ename: null**

**_JSPService()**

**config: org.apache.catalina.core.StandardWrapperFacade@8cb492**

**this: org.apache.jsp.Test2_JSP@1a2edef**

**ename: null**



**<web-app>**

```
    <servlet>
        <servlet-name>test</servlet-name>
        <jsp-file>/Test2.jsp</JSP-file>
        <init-param>
            <param-name>ename</param-name>
            <param-value>ramchandra</param-value>
        </init-param>
    </servlet>
    <!-- <servlet-mapping>
        <servlet-name>test</servlet-name>
        <url-pattern>/Test2.jsp</url-pattern>
    </servlet-mapping> -->
</web-app>
```

==========

**JSP Tags:**

---

We have 5 types of tags

1. Scripting tags

**2. Directive tags (page, include, taglib)**

**3. Standard action tags**

**4. Custom action tags**

**5. JSTL**

**Scripting Tags: <%------%>**

**Comments in jsp:**

**1. html comment <!-- comment text -->**

**This tag is not ignored by the JSP container, this tag will be placed into PageImplementationclass (PIC), handover to browser by Servlet container, but browser will ignores.**

**2. JSP commnet <%-- commnet text --%>**

**This tag is ignored by the Jsp Container**

**3. Java comment: If we are writing java comments outside of html comment and jsp comment these comments will be treated as html code.**

**If we are writing java comments in declarative and scriptlet tag, expression tags, these comments not ignored by the jsp container these comments are placed into PIC in translation phase, but in compilation jsp container ignores.**

**<%= //we can't %>**

**The above statement is invalid the reason without printing any value we can't write comment.**

**<%="cant" //we can %>**

**The above statement is valid.**

```
<html>
<body bgcolor='yellow'>
	<h1>THIS IS JSP PAGE</h1>
	<!-- this is html comment -->
	<%-- this is jsp comment --%>
	//this is java comment
	/* this is java ml comment*/
	/** this is java d comment*/
	<% //this java comment in ST
	%>
	<%!
		//this is declarative tag
	%>

</body>
</html>
```

## 4.declaration tag <%! ----%>(class level coding)

### Example on declaration tag:

```
	<html>
<body bgcolor='yellow'>
	<%!
		static int m1(int x){
			System.out.println("this is static m1 method");
			if(x >= 0){
				System.out.println("x is positive number");
			}
			else
				System.out.println("x is negative number");
```

```
                return 111;
          }
     %>
     <%!  static int a = m1(123); %>
<%!  static int a = m1(new java.util.Scanner(System.in).nextInt()); %>
</body>
</html>
```

## 5. Scriplet tag <% --- %>  (_JSPService(-,-))

## 6. Expression tag <%= ---- %>(_JSPService(-,-))

## Example on scriptlet and expression tags:

```
<html>
<body bgcolor='yellow'>
     <%
          String s = "Nationalization";
          if(s.contains("ali")){
               System.out.println("if block executes ali is there");
          }
          else
               System.out.println("if block executes ali is not
there");
     %>
     <%
          String s1 = "Internationalization";
          String result="";
          result=s1.substring(5, 11);
          System.out.println("Result: "+result);
     %>
     <%= result %>

</body>
</html>
```

## Example on above tags:

## <!--THIS IS HTML COMMENT -->

```
This is html code </br>
//THIS IS JAVA COMMENT
THIS IS JSP COMMENT
<%!
	static{
		System.out.println("static block");
	}
%>
<%!
	{
	System.out.println("non-static block");
	}
%>
<%!
	public Test3_JSP(){
		System.out.println("constructor");
	}
%>
<%!System.out.println("we cannot do this"); %>
<%!
```

```
        static int m1(){
        System.out.println("static m1 method");
        return 8989;
        }
%>
<%!
        int x = 111;
        static int y = 333;
        private int z = 444;
        protected int l = 555;
        transient int m = 666;
        public int n = 777;
        static int o = m1();
%>
<%
    int aa1 = 111;
        final String aa2 = "ram";
        m1();
%>
<%= o %>
```

**<%= "System.out.println"%>**

**<%= m1() %>**

**Errors in Jsp:**

**We have three types of errors in JSP**

**a. Translation Errors:**

**The errors which are given by the jsp container due to bad syntax of tags. If translation error occurs jsp container not creates .java file and display 500 error on browser.**

**< % %> --> this will be treated as html code**

**<% --> Translation Error**

**%> --> html code**

**<% % > --> Translation Error**

**B.Compile Time Errors:**

**If compiler unable understand we will get compile time error.**

**In this time jsp container will creates .java file but not .class file.**

**<% ! %> --> Compile time Error**

**<% <% %>--> Compile time Error**

**<% int a = 10 %> --> Compiler time Error**

**<% %> %> --> html code**

**C.Execution Error:**

**If there is any problem with logic, then we will get runtime error.**

**In the time jsp container will creates .java and .class file but servlet**

**container will give errors.**

**<% int a = 10/0; %>**

**<% int a[] = {11,22,33};**

**System.out.println(a[3]);**

**%>**

**Communication between one .jsp file to other .jsp, .html,.java,.txt file:**

For doing above process we required pageContext.foraward(-).

Test.jsp:

```
<html>
<body bgcolor='yellow'>
<%-- <% pageContext.forward("./Test1.jsp"); %> --%>
<%-- <% pageContext.forward("./Test2.html"); %> --%>
<%-- <% pageContext.forward("./Test3.txt"); %> --%>
<% pageContext.forward("./ts"); %>
</body>
```

```
</html>
```

Test1.jsp:

```
<html>
<body bgcolor='yellow'>
     <h1>THIS IS Test1.jsp FILE</h1>
</body>
</html>
```

Test2.html:

```
<html>
<body bgcolor='yellow'>
     <h1>THIS IS Test.html FILE</h1>
</body>
</html>
```

Test3.txt:

**THIS IS NORMAL TEXT FILE**

**TestServlet.java**

```
import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.GenericServlet;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

public class TestServlet extends GenericServlet{

@Override

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

```
        res.setContentType("text/html");        PrintWriter out
= res.getWriter();        out.println("<body bgcolor='yellow'>");
    out.println("THIS IS TESTSERVLET");
    out.println("</body>");

    }
}
```

**Implicit Objects:**

**The objects which are created or given by the container to implement our jsp logic are called Implicit Objects.**

**We have 9 implicit objects.**

| | |
|---|---|
| **1. config** | **ServletConfig** |
| **2. session** | **HttpSession** |
| **3. application(context)** | **ServletContext** |
| **4. pageContext** | **PageContext** |
| **5. out** | **JspWriter** |
| **6. page** | **java.lang.Object** |
| **7. exception** | **java.lang.Throwable** |
| **8. request** | **HttpServletRequest** |
| **9. response** | **HttpServletResponse** |

**Here out is not belongs PrintWriter it is belongs to JspWriter.**

**With the help of page implicit object we can't call any variable and method within jsp page.**

**The reason is page object belongs to java.lang.Object.**

**It is suitable for calling method from Object class.**

```
<html>
    <body bgcolor='yellow'>
        <form>
            <%int a = 111; %>
            <%! void m1(){
                System.out.println("m1 method");
            } %>
            <%= a %>
            <%= page %>
            <%= page.a %>  //invalid
            <%= page.m1() %>     //invalid
            <%= page.toString() %>

        </form>
    </body>
</html>
```

**Where can we use implicit objects in jsp page?**

**We can use jsp implicit object only within two tags**

a. **Expression tag**
b. **Scriptlet tag**

**That means these implicit object are only available in _jspService(-,-). We can't use within the jspInit() and jspDestroy().**

```
<html>
```

```
    <body bgcolor='yellow'>
        <form>
            <%=application %><br/>
            <%=config %><br/>
            <%=session %><br/>
            <%=pageContext %><br/>
            <%=out %><br/>
            <%=page %><br/>
            <%=request %><br/>
            <%=response %><br/>
            <%-- <%=exception %> --%>
            <%
            System.out.println(application);
            %>
        <%! public void jspInit(){
            System.out.println(application);
            System.out.println(config);
        } %>
        <%! public void jspDestroy(){
            System.out.println(application);
            System.out.println(config);
        } %>
        </form>
    </body>
</html>
```

In the above we need to comment jspInit() and jspDestroy() statements, otherwise we will get compile time errors.

How can we use config and context implicit objects in jspInit() and jspDestroy()?

To interact with config and context objects within jspInit() and jspDestroy() we required factory methods like

a.getServletContext()
b.getServletConfig()

In _jspService(-,-) either we can use factory method or implicit objects

a.getServletContext()
b.getServletConfig()
c.config
d.application

```
<html>
    <body bgcolor='yellow'>
        <form>
            <%
        ServletConfig sconfig = getServletConfig();
    ServletContext scontext = getServletContext();
            System.out.println("sconfig: "+sconfig);
        System.out.println("scontext: "+scontext);
            %>
            <%= config %>
            <%= application %>
        <%! public void jspInit(){
        ServletConfig sconfig = getServletConfig();
    ServletContext scontext = getServletContext();
            System.out.println("sconfig: "+sconfig);
        System.out.println("scontext: "+scontext);
        } %>
        <%! public void jspDestroy(){
        ServletConfig sconfig = getServletConfig();
    ServletContext scontext = getServletContext();
            System.out.println("sconfig: "+sconfig);
        System.out.println("scontext: "+scontext);
        } %>
        </form>
    </body>
</html>
```

---

Web.xml:

```
<web-app>
        <context-param>
            <param-name>password</param-name>
            <param-value>chandra</param-value>
        </context-param>
    <servlet>
```

```
        <servlet-name>ts</servlet-name>
        <jsp-file>/Test.jsp</jsp-file>
        <init-param>
            <param-name>user</param-name>
            <param-value>ram</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>ts</servlet-name>
        <url-pattern>/Test.jsp</url-pattern>

    </servlet-mapping>

</web-app>
```

Test.jsp:

```
<html>
    <body bgcolor='yellow'>
        <form>
            <%
        ServletConfig sconfig = getServletConfig();
    ServletContext scontext = getServletContext();
    System.out.println("sconfig: "+sconfig);
    System.out.println("user:
"+sconfig.getInitParameter("user"));
            System.out.println("scontext:
"+scontext);
    System.out.println("password:
"+scontext.getInitParameter("password"));
            %>
            <%= config.getInitParameter("user") %>
            <%=
application.getInitParameter("password") %>
        <%! public void jspInit(){
```

```
            ServletConfig sconfig =
getServletConfig();
            ServletContext scontext =
getServletContext();
            System.out.println("sconfig: "+sconfig);
        System.out.println("scontext: "+scontext);
            System.out.println("user:
"+sconfig.getInitParameter("user"));

            System.out.println("password:
"+scontext.getInitParameter("password"));
        } %>
        <%! public void jspDestroy(){
            ServletConfig sconfig =
getServletConfig();
            ServletContext scontext =
getServletContext();
            System.out.println("sconfig: "+sconfig);
            System.out.println("user:
"+sconfig.getInitParameter("user"));
            System.out.println("scontext:
"+scontext);
            System.out.println("password:
"+scontext.getInitParameter("password"));
        } %>
        </form>
    </body>
</html>
```

Q) Can we able create variable names like confing and application in jspInit() , jspDestroy(), _jspService()?

We can able to create variable names like config, application, page … in jspInit() and jspDestroy(), in these methods these names are not treated as implicit object the reason is by default implicit object are not available to jspInit() and jspDestroy().

We can't create variable names like config, application, page … in _jspService(-,-). The on top of these name container already created implicit objects and placed into _jspService(-,-). If we are trying to create we will compiletime error like duplicate variable.

```
<html>
    <body bgcolor='yellow'>
        <form>
        <%! public void jspInit(){
        System.out.println("jspInit()");
        String config = "This is config variable";
String application = "This is application variable";
                System.out.println(config);
                System.out.println(application);
            }
        %>
        <%! public void jspDestroy(){
            System.out.println("jspDestroy");
            String config = "This is config
variable";
            String application = "This is
application variable";
            System.out.println(config);
            System.out.println(application);
        } %>
```

```
    //the following code invalid
    <%
        String config = "this is config vairable";
String application = "This is application variable";
        %>
        </form>
    </body>
</html>
```

Q) Can we use exception implicit object directly in jsp?

A) No.

Q) How can we use exception implicit object in jsp?

A) Yes, with the help of directive tag.

All 8 implicit object we can use directly exception “exception” object, the reason by default jsp container unable recognize the “exception” implicit object.

If we want to use we need declare one directive tag like bellow.

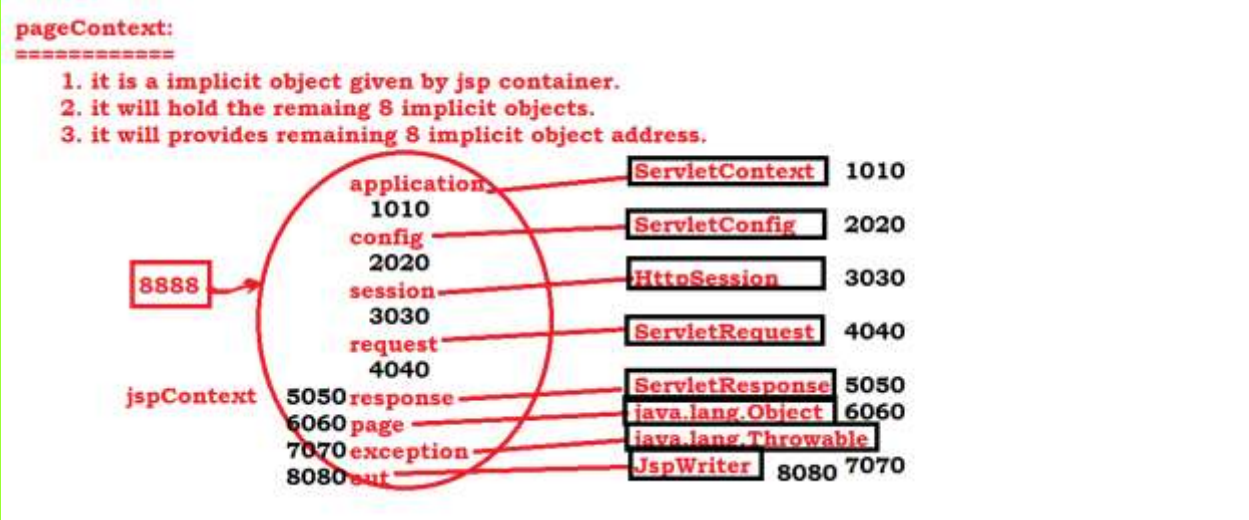
```
<%@ page isErrorPage="true" %>
<html>
    <body bgcolor='yellow'>
        <form>
        <%@ page isErrorPage="true" %>
        <%= application %>
        <%= config %>
        <%= exception %>
        </form>
    </body>
```

```
</html>
```

IN the above exception implicit object will gives value like null.

The reason there is no exception in this page.

**PageContext:**




**Usages:**

a. **Getting implicit objects.**
b. **Storing/ retrieving / deleting the data from scope objects.**
c. **Making communication with other .jsp file,.html file, .java file,.txt file.**

**a. Getting implicit objects:**

```
<html>
    <body bgcolor='yellow'>
```

```
            <form>
            <%!
        public void m1(PageContext pc){
            ServletRequest sreq = pc.getRequest();
            ServletResponse sres = pc.getResponse();
        ServletContext scon = pc.getServletContext();
        ServletConfig scfg = pc.getServletConfig();
            HttpSession hse = pc.getSession();
            java.lang.Object obj = pc.getPage();
        java.lang.Throwable thr = pc.getException();
            JspWriter jw = pc.getOut();
                }
            %>
            <%
    ServletRequest sreq = pageContext.getRequest();
    ServletResponse sres = pageContext.getResponse();
ServletContext scon =pageContext.getServletContext();
ServletConfig scfg = pageContext.getServletConfig();
    HttpSession hse = pageContext.getSession();
        java.lang.Object obj = pageContext.getPage();
java.lang.Throwable thr = pageContext.getException();
            JspWriter jw = pageContext.getOut();
            m1(pageContext);
            %>
        </form>
    </body>
</html>
```



**b. Storing/ retrieving / deleting the data from scope objects.**

**1. setAttribute(-,-):**

**To place the data page scope object**

**2. setAttribute(-,-,-):**

**To place the data specific scope object**

3. **getAttribute(-):**
   **Read the data from page scope object**
4. **getAttribute(-,-):**
   **Read the data from specific scope object. In the above methods name is not available, it will returns null.**
5. **findAttribute(-)**
   **Read the data from page scope object if not available control goes to request scope object if not available control goes to session scope object if not available control goes to application scope object. If not null value**
6. **removeAttribute(-)**
   **It will remove the data from all scope objects.**
7. **removeAttribute(-,-)**
   **It will useful for removes the data from one specified given scope.**

```
<html>
    <body bgcolor='yellow'>

        <form>
        <%pageContext.setAttribute("sno", 101); %>
        <%pageContext.setAttribute("sname", "ram");%>
        <%pageContext.setAttribute("sfee",5000); %>
    sno: <%=pageContext.getAttribute("sno") %><br/>
sname: <%=pageContext.getAttribute("sname") %><br/>
sfee: <%=pageContext.getAttribute("sfee") %><br/>
```

```
<%pageContext.setAttribute("eno", 5678,
PageContext.REQUEST_SCOPE); %>

eno: <%=pageContext.getAttribute("eno") %><br/>

eno:
<%=pageContext.getAttribute("eno",PageContext.REQUEST
_SCOPE) %><br/>
    <%pageContext.removeAttribute("sno"); %>
    sno: <%=pageContext.getAttribute("sno") %><br/>
        </form>
    </body>
</html>
```

---

**<%pageContext.setAttribute("sno", 101); %>**

**<!--IN the above statement sno will be placed into page scope object  -->**

**<%pageContext.setAttribute("sname", "ram"); %>**

**<%pageContext.setAttribute("sfee",5000); %>**

**sno: <%=pageContext.getAttribute("sno") %><br/>**

**<!-- In the above statement sno will be read from page scope object -->**

**sname: <%=pageContext.getAttribute("sname") %><br/>**

```
                sfee: <%=pageContext.getAttribute("sfee")
%><br/>

                <%pageContext.setAttribute("eno",
5678,PageContext.REQUEST_SCOPE); %>

                eno: <%=pageContext.getAttribute("eno")
%><br/>

                eno:
<%=pageContext.getAttribute("eno",PageContext.REQU
EST_SCOPE) %><br/>

                <%pageContext.setAttribute("sno",
201,pageContext.REQUEST_SCOPE); %>

                <%pageContext.removeAttribute("sno");
%>

<!-- IN the above statement sno will be deleted from all
scope object -->

                sno from page:
<%=pageContext.getAttribute("sno") %><br/>

                sno from request:
<%=pageContext.getAttribute("sno",PageContext.REQU
EST_SCOPE) %><br/>

<% pageContext.setAttribute("cname",
"mca",PageContext.APPLICATION_SCOPE); %>

            cname from page:
<%=pageContext.getAttribute("cname") %><br/>

cname from request:
```

**<%=pageContext.getAttribute("cname",PageContext.REQUEST_SCOPE) %><br/>**

**cname from application:**

**<%=pageContext.getAttribute("cname",PageContext.APPLICATION_SCOPE) %><br/>**

**<%=pageContext.findAttribute(") %>**

---

```
<html>
    <body bgcolor='yellow'>

        <form>

<%pageContext.setAttribute("cname", "mca1"); %>
<%pageContext.setAttribute("cname",
"mca2",PageContext.REQUEST_SCOPE);
%>
<%pageContext.setAttribute("cname",
"mca3",PageContext.SESSION_SCOPE);
%>

<%pageContext.setAttribute("cname", "mca4",
            PageContext.APPLICATION_SCOPE); %>
    <%-- <%pageContext.removeAttribute("cname"); %> -
-%>
    <%pageContext.removeAttribute("cname",PageContext
.REQUEST_SCOPE); %>
CINEMA NAME p:<%=pageContext.getAttribute("cname")
%><br/>
CINEMA NAME r:<%=pageContext.getAttribute("cname",
        PageContext.REQUEST_SCOPE) %><br/>
```

```
CINEMA NAME s:<%=pageContext.getAttribute("cname",
        PageContext.SESSION_SCOPE) %><br/>
CINEMA NAME a:<%=pageContext.getAttribute("cname",
        PageContext.APPLICATION_SCOPE) %><br/>

        </form>
    </body>
</html>
```

**WARNING:**

**The basic mistake doing by the programmer is placing the data in one scope object and reading that data from another scope object.**

---

**C. Making communication with other .jsp file, .html file, .java file,.txt file.**

**→ forward(-)**

**→include(-)**

**Diagram on pageContext forward(-):**

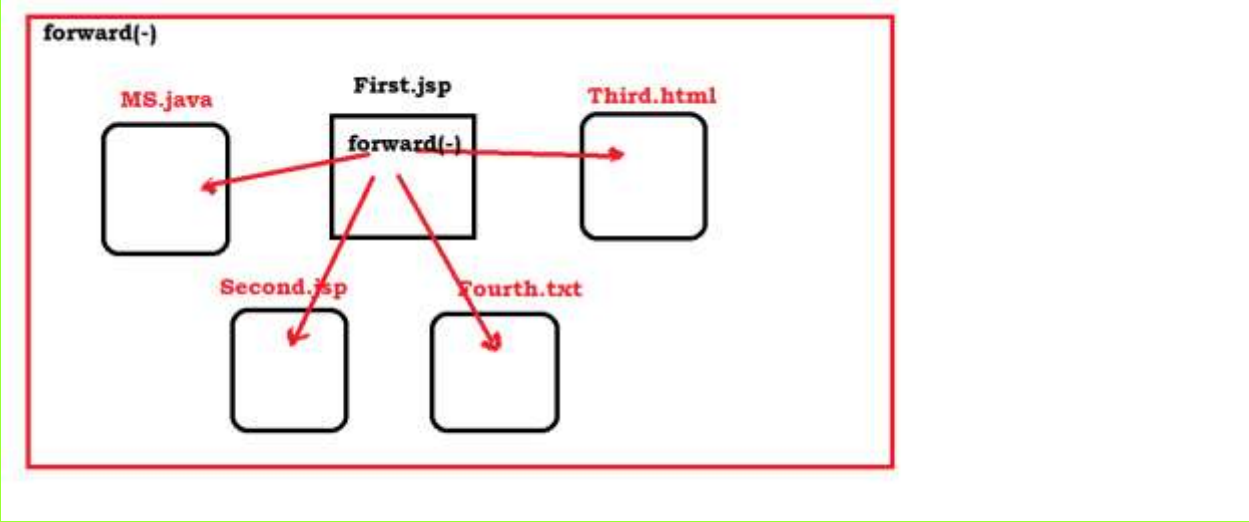
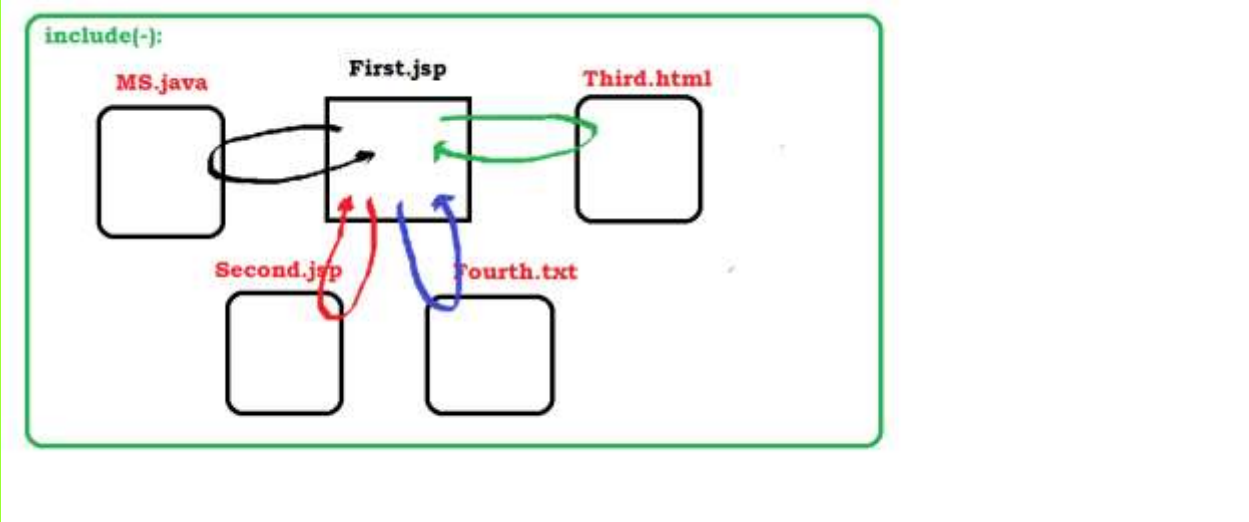


**Diagram on pageContext include(-):**




**First.jsp:**

```
<html>
    <body bgcolor='yellow'>
    THIS IS FIRST.JSP FILE
    <form>
<%-- <%pageContext.forward("./Second.jsp"); %> --%>
<%-- <%pageContext.forward("./Third.html"); %> --%>
<%-- <%pageContext.forward("./Fourth.txt"); %> --%>
<%-- <%pageContext.forward("./ms"); %> --%>
<%-- <%pageContext.include("./Second.jsp"); %>
<%pageContext.include("./Third.html"); %>
<%pageContext.include("./Fourth.txt"); %> --%>
        <%pageContext.include("./ms"); %>
    </form>
    </body>
</html>
```

**Second.jsp:**

```
<html>
    <body bgcolor='yellow'>
    THIS IS SECOND.JSP FILE <br/>
    </body>
</html>
```

Third.html:

```
<html>
    <body bgcolor='yellow'>
    THIS IS THIRD.HTML FILE <br/>
    </body>
</html>
```

Fourth.txt:

```
<html>
    <body bgcolor='yellow'>
    THIS IS FOURTH.TXT FILE
    </body>
</html>
```

MyServlet.java:

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

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class MyServlet extends GenericServlet{

    @Override
```

```
    public void service(ServletRequest request,
ServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out  = response.getWriter();
        out.println("<body bgcolor='yellow'>");
        out.println("THIS IS SERVLET CLASS");
        out.println("</body>");
    }
}
```

web.xml:

```
<web-app>
    <servlet>
        <servlet-name>ms</servlet-name>
        <servlet-class>MyServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ms</servlet-name>
        <url-pattern>/ms</url-pattern>
    </servlet-mapping>
</web-app>
```

**Scope Objects:**

**We have 4 scope objects.**

**These are useful carry the information from .jsp file to another .jsp file.**

- **page**
- **request**
- **application (context)**
- **session**

**In all the scope object application scope object having highest accessibility that means we can that scope object information from all browser as well windows also.**

**Session scope object data can be accessible within the same browser but all windows.**

**Request scope object data can be accessible only one window, once response handover to browser request scope object data will be destroyed.**

**Page scope object data can be accessible only within the same page.**



**For retrieving JSP implicit objects.**

```
<%="hi" %><br/>

<%=request %><br/>
```

**<%=response %><br/>**

**<%=application %><br/>**

**<%=session %><br/>**

**<%=page %><br/>**

**<%=pageContext %> <br/>**

**<%=out%><br/>**

**<%=config%><br/>**

**<%= config.getInitParameter("ename") %>**

**<%-- <%=exception %> invalid --%>**

**The above 9 objects are not working in JspInit() and JSPDestroy().**

**If we want to working with them we need to write the code for getting those objects.**

```
<%!public void jspInit(){
System.out.println("init method");
//System.out.println(config);
ServletConfig cfg = getServletConfig();
System.out.println("cfg: "+cfg);
ServletContext cxt = getServletContext();
System.out.println("cxt: "+cxt);
} %>
```

**<%!**

```
    public void jSPDestroy(){
        //System.out.println(config);
        System.out.println("this is JSPdestroy()");
        ServletConfig cfg = getServletConfig();
       System.out.println("cfg: "+cfg);
       }
%>
```

**For strong and retrieving and removing attributes from JSP 4 scope objects.**

```
hi
<%
pageContext.setAttribute("eno1", 7279);
%>
<%
pageContext.setAttribute("eno2", 7280);
%>
<%
pageContext.setAttribute("eno3", 7281);
%>
<%
pageContext.setAttribute("eno4", 7282);
%>
eno1:<%=pageContext.getAttribute("eno")%><br/>
eno1:<%=pageContext.getAttributesScope("eno1")%><br/>
eno2:<%=pageContext.getAttributesScope("eno2")%><br/>
eno3:<%=pageContext.getAttributesScope("eno3")%><br/>
eno4:<%=pageContext.getAttributesScope("eno4")%><br/>
eno4:<%=pageContext.getAttributesScope("eno5")%><br/>
<%pageContext.setAttribute("ename1", "ram1",PageContext.PAGE_SCOPE);
%>
<%pageContext.setAttribute("ename2",
"ram2",PageContext.REQUEST_SCOPE); %>
<%pageContext.setAttribute("ename3",
"ram3",PageContext.SESSION_SCOPE); %>
<%pageContext.setAttribute("ename4", "ram4",PageContext.
```

```
APPLICATION_SCOPE); %>
ename1:<%=pageContext.getAttribute("ename1") %><br/>
ename1:<%=pageContext.getAttribute("ename1",PageContext.PAGE_SCOPE)
%><br/>
ename1:<%=pageContext.getAttribute("ename1",PageContext.REQUEST_SCOPE)
%><br/>
ename1:<%=pageContext.getAttribute("ename1",PageContext.SESSION_SCOPE)
%><br/>
ename1:<%=pageContext.getAttribute("ename1",PageContext.APPLICATION_SC
OPE) %><br/>
ename2:<%=pageContext.getAttribute("ename2",PageContext.REQUEST_SCOPE)
%><br/>
ename3:<%=pageContext.getAttribute("ename3",PageContext.SESSION_SCOPE)
%><br/>
ename4:<%=pageContext.getAttribute("ename4",PageContext.APPLICATION_SC
OPE) %><br/>
<%pageContext.removeAttribute("ename1"); %>
ename1:<%=pageContext.getAttribute("ename1") %><br/>
ename1:<%=pageContext.getAttribute("ename1",PageContext.PAGE_SCOPE)
%><br/>
bye
```

The common mistake is doing by the programmer is placing data in one scope and reading the data in another scope, then we are always getting null values. If we are doing on null values we will get NullPointerException.

Page scope object only within the same JSP file, it is not sharable between other JSP files.

Test4.jsp:

```
<% application.setAttribute("eno",1234);%>
<% session.setAttribute("ename","ram");%>
<% request.setAttribute("dept","faculty");%>
<% pageContext.forward("./Test5.jsp");%>
```

## Test5.jsp:

```
THIS IS TEST5.jsp FILE.....
<%= application.getAttribute("eno")%><BR/>
<%= session.getAttribute("eno")%><BR/>
```

```
<%= request.getAttribute("eno")%><BR/>
<%-- <%= page.getAttribute("eno")%> --%>
<%= session.getAttribute("ename")%><BR/>
<%= request.getAttribute("dept")%><BR/>
<%= application.getAttribute("dept")%><BR/>
<%= request.getAttribute("ename")%><BR/>
END OF TEST5.jsp FILE
```

## Special Test case on applicaton scope object:

## Test4.jsp:

```
<% application.setAttribute("eno",1234);%>

<% pageContext.forward("./Test5.jsp");%>
```

## Test5.jsp:

```
THIS IS TEST5.jsp FILE......
<%= application.getAttribute("eno")%><BR/>

END OF TEST5.jsp FILE
```

Call Test5.jsp file from different browser directly

We will get value like 1234. This scope object will share by the enduser(browsers).

## Special Test case on session scope object:

## Test4.jsp:

```
<% session.setAttribute("eno",12345);%>

<% pageContext.forward("./Test5.jsp");%>
```

## Test5.jsp:

```
THIS IS TEST5.jsp FILE......
<%= session.getAttribute("eno")%><BR/>

END OF TEST5.jsp FILE
```

Call Test5.jsp file from different browser directly first we will get null value.

Again call Test4.jsp from that browser then we will get 12345 values.

Call Test5.jsp file from in the browser directly first we will get 12345 values.

This scope object will share only one end user (browser).

## Special Test case on request scope object:

## Test4.jsp:

```
<% request.setAttribute("eno",1234567);%>

<% pageContext.forward("./Test5.jsp");%>
```

## Test5.jsp:

```
THIS IS TEST5.jsp FILE......
<%= request.getAttribute("eno")%><BR/>

END OF TEST5.jsp FILE
```

Call Test5.jsp file from different browser directly first we will get null value.

Whenever we making a request Test5.jsp file servlet container will create neq request object with empty values.

Again call Test4.jsp from that browser then we will get 1234567 values.

Again open one more tab in the browser and call Test5.jsp we will get null only.

This scope object only shareable between one request and response cycles.

## Page Scope objects:

## Test4.jsp:

```
<% pageContext.setAttribute("eno",1234567);%>
<%= pageContext.getAttribute("eno")%><BR/>
```

```
<% pageContext.forward("./Test5.jsp");%>
```

**Test5.jsp:**

```
<%= pageContext.getAttribute("eno")%><BR/>
```

The parameter and value or different value (attributes) we place in different scope objects.

Within the one scope object we can able place multiple attributes.

We can also delete attributes from the scope objects.

**How to read data from html in jsp:**

**UserDetails.html:**

```
<html>
    <body bgcolor='gold'>
        <form action='./ud.jsp'>
        Enter Your Name: <input type='text'
name="uname"/><br/>
        Enter Your age : <input type='password'
name='uage'/><br/>
        Enter Your course: <input type='text'
name="ucourse"/><br/>
            <input type='submit' value='SUBMIT'/>

        </form>
    </body>

</html>
```

**Ud.jsp:**

```
<html>
```

```
    <body bgcolor='gold'>
        <%
            String uname =
request.getParameter("uname");
            String uage =
request.getParameter("uage");
            String ucourse =
request.getParameter("ucourse");
        %>
        <%= uname %>
        <% out.println(uage); %>
        <%= ucourse %>
    </body>
</html>
```

**Test.html:**

```
<html>
    <body bgcolor='yellow'>
        <h1>THIS IS HTML FILE</h1>
        <form action='./Test4.jsp'>
            Enter your name: <input type='text' name='name'/>
            <input type='submit' name='reqtype' value='submit'/>
        </form>
    </body>
</html>
```

**Test4.jsp:**

```
<body bgcolor='yellow'>
    <%String name = request.getParameter("name"); %>
    Hi...<%=name %>

</body>
```

In the above program only response like Hi… ramchandra committed but html text field not committed as response.

So end user needs to click on back button, then only get the html text field for entering new value.

To overcome this problem we we need to write again html code in .jsp file like bellow.

```
<body bgcolor='yellow'>
      <%String name = request.getParameter("name"); %>
      Hi...<%=name %>
      <form action='./Test4.jsp'>
                  Enter your name: <input type='text' name='name'/>
                  <input type='submit' name='reqtype' value='submit'/>
            </form>

</body>
```

In above approach .html code repeated.

To overcome the above problem write html code in .jsp file send the request to .jsp directly.

Test4.jsp:

```
<html>
      <body bgcolor='yellow'>
            <h1>THIS IS HTML FILE</h1>
            <%String name = request.getParameter("name"); %>
            Hi...<%=name %>

            <form action='./Test4.jsp'>
                  Enter your name: <input type='text' name='name'/>
                  <input type='submit' name='reqtype' value='submit'/>
            </form>
      </body>
</html>
```

In the above approach we are facing one more problem like, in the first request we are getting null by request. getParameter().

From second request onwards only we will get valid answer.

To overcome this problem we need write the code in the following manner.

Test4.jsp:

```
<html>
```

```
	<body bgcolor='yellow'>
	<form action='./Test4.jsp'>
		Enter your name: <input type='text' name='name'/>
		<input type='submit' name='reqtype' value='submit'/>
	</form>
	<%String name = request.getParameter("name");
		if(name==null){
			return;
		}
	%>
	HI...<%= name %>

	</body>
</html>
```

## Develop a program to read two values for addition:

Test4.jsp:

```
<html>
	<body bgcolor='yellow'>
	<form action='./Test4.jsp'>
		Enter First Value: <input type='text'
name='fno'/><br/>
		Enter Second Value: <input type='text'
name='sno'/><br/>
		<input type='submit' name='add' value='ADD'/>
	</form>
	<%String fno = request.getParameter("fno");
	String sno = request.getParameter("sno");
		if(fno==null || sno ==null){
			return;
		}
	%>
	<% int no1 = Integer.parseInt(fno);
		int no2 = Integer.parseInt(sno);
	%>
	Addition is:<%= no1+no2 %>

	</body>
</html>
```

How to develop bellow calculation program:


```
<html>
      <body bgcolor='yellow'>
      <form action='./Test4.jsp'>
                  Enter First Value: <input type='text'
name='fno'/><br/>
                  Enter First Value: <input type='text'
name='sno'/><br/>
                  <input type='submit' name='reqtype' value='ADD'/>
                  <input type='submit' name='reqtype' value='SUB'/>
                  <input type='submit' name='reqtype' value='MUL'/>
                  <input type='submit' name='reqtype' value='DIV'/>
            </form>
            <%String result="";
            String fno = request.getParameter("fno");
            String sno = request.getParameter("sno");
                  if(fno==null || sno ==null){
                        System.out.println("if block");
                        fno=" ";
                        sno=" ";
                  }
                  else{
                        String type = request.getParameter("reqtype");

                        int no1 = Integer.parseInt(fno);
                        int no2 = Integer.parseInt(sno);
                        if(type.equalsIgnoreCase("add")){
                              result = "Result is: "+(no1+no2);
                        }
                        else
                              if(type.equalsIgnoreCase("sub")){
                                    result = "Result is: "+(no1-no2);
```

```
                        }
                        else if(type.equalsIgnoreCase("mul")){
                              result = "Result is: "+(no1*no2);
                        }
                        else if(type.equalsIgnoreCase("div")){
                              result = "Result is: "+(no1/no2);
                        }
                  }
            %>
            <%= result %>

      </body>
</html>
```

**Directive tags:**

**If we want provide information to jsp container about PIC creation according to our requirement (jsp author) we should go for directive tags.**

**a. page**

**b. taglib**

**c. include**

**page is used for in the following places**

- **Import any predefine classes**
- **Extends PIC from our own class**
- **Create JSP as thread safe**
- **Disable the session**
- **Set the content other than "text/html"**
- **Handle the exceptions**

**If we are not using page directive tag, jsp container will create PIC with default code that is**

a. **Import only jsp and servlet import statements.**
b. **PIC Extends server vendor given super class (HttpJspBase).**
c. **With no thread safty.**
d. **With session**
e. **With default context type “text/html”**
f. **By default exception implicit object disable.**

**We can declare directive tags any where is .jsp file, if we declare any where jsp container will those directive tags in some required places.**

**Better to use directive tags in the top of the jsp.**

**Page directive having the following attributes:**

**<%@ page**

**language**

**import**

**extends**

**isThreadSafe**

**contentType**

**pageEncoding**

**session**

**errorPage**

**isErrorPage**

**autoFlush**

**buffer**

**info**

**isELIgnored**

**%>**

---

```
<html>
    <body bgcolor='yellow'>
        <form>
            <%@ page
import="java.util.ArrayList,java.util.Vector" %>
            <%@ page import='java.util.Scanner' %>
            <%-- <%@ page import='java.util.*' %> --
%>
            <%@ page session='false' %>
            <%@ page
contentType="application/msword" %>
            <%
                ArrayList al = new ArrayList();
                Scanner scan = new
Scanner(System.in);
                Vector v = new Vector();
            %>
            <%= "good" %>
            <%-- <%= session %> --%>
        </form>
    </body>
</html>
```

**********

**By default container will processing multiple requests at a time. If want to processing the single request at a time we need to implements SingleThreadModel interface we can do this with the support of the following tag.**

```
<html>
    <body bgcolor='yellow'>
        <form>
            <%= session %>
            <%= exception %>
            <%@ page isErrorPage="true" %>
            </form>
    </body>
</html>
```

If we want use exception implicit object in .jsp we should write this attribute, that is isErrorPage=”true”

We can write this directive before or after implicit object usage statement.

Working with isErrorPage and errorPage:

One.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <%
                try{
                    int a = 10/0;
                }catch(ArithmeticException e){
out.println("<h2>Don’t enter zero as denominator
</h2>");
                }
            %>
    </form>
    </body>
</html>
```

Two.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <%
                try{
                    int a = Integer.parseInt("a");
                }catch(NumberFormatException e){
out.println("<h2>Provide integer type values</h2>");
                }
            %>
        </form>
    </body>
</html>
```

Three.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <%
                try{
                    out.println("ram".charAt(3));

    }catch(StringIndexOutOfBoundsException e){
                    out.println("<h2>There is no
sufficient index character</h2>");
                }

            %>
        </form>
    </body>
</html>
```

In the above program in all .jsp we are exception handling code by using try and catch blocks.

To avoiding exception handling and maintain exception message in separate jsp we should use isErrorPage attribute.

**One.jsp**

```
<html>
    <body bgcolor='gold'>
        <form>
            <%@  page errorPage="Error.jsp" %>
            <%  int a = 10/0;%>
        </form>
    </body>
```

```
</html>
```

Two.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <%@ page errorPage="Error.jsp" %>
            <%      int a=10;
                    int b = 0;
                    int c = a/b;
            %>
        </form>
    </body>
</html>
```

Whenever we using errorPage attribute jsp container will create separate .java files and .class files for the .jsp files like bellow.

One_jsp.java

One_jsp.class

Error_jsp.java

Error_jsp.class

**Error.jsp:**

```
<h1>DONT WRITE ZERO AS DENOMINATOR</h1>
```

Three.jsp:

```
<html>
    <body bgcolor='gold'>
```

```
        <form>
            <%@ page errorPage="Error.jsp" %>
            <%  out.println("ram".charAt(3));%>
        </form>
    </body>
</html>
```

**In the One.jsp and Two.jsp one exception object is raises that are ArithmeticException. We are forwarding that exception object to Error.jsp and print "don't enter zero denominators.**

**In above case exception object and message are suitable but Three.jsp exception and Error.jsp file exception message are not match. That means in above we are unable to handle multiple exceptions by using one .jsp file to achieve this requirement we should go for "isErrorPage" attribute.**

**One.jsp:**

```
<html>
    <body bgcolor='gold'>
        <form>
            <%@  page errorPage="Error.jsp" %>
            <%  int a = 10/0;%>
        </form>
    </body>
</html>
```

**Two.jsp:**

```
<html>
    <body bgcolor='gold'>
        <form>
```

```
            <%@ page errorPage="Error.jsp" %>
            <%  int a = Integer.parseInt("a");
            %>
        </form>
    </body>
</html>
```

Three.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <%@ page errorPage="Error.jsp" %>
            <%  out.println("ram".charAt(3));%>
        </form>
    </body>
</html>
```

Error.jsp:

```
<body bgcolor='gold'>
<%@ page isErrorPage="true"%>
<%
    try{
        throw exception;
    }
    catch(ArithmeticException ae){
        out.println("<h2>dont enter zero as
denominator</h2>");
    }
    catch(StringIndexOutOfBoundsException ae){
        out.println("<h2>there is no sufficient
character to print</h2>");
    }
    catch(NumberFormatException e){
```

```
        out.println("<h2>please enter numeric type
data</h2>");
    }
%>
</body>
```

**11.jsp:**

**<%@ page isThreadSafe='false' %>**

**<%= this is JSP file %>**

**----------**

**<%@ page session='false' %>**

**With the support above tag we can avoid session object in JSP.**

**<%@ page session='fslse' %>**

**<%=session%> //invalid**

**<%="good"%>**

---

```
<html>
<body bgcolor='yellow'>
    <%-- <%@ page isThreadSafe="false" session="true"
import='java.util.*' %> --%>
    <%@ page import= 'java.io.*'%>
    <%@ page import= 'java.util.*'
extends='javax.servlet.GenericServlet'%>
```

```
    <h1>THIS IS TEST1 JSP FILE-WITH SESSION
OBJECT</h1>
    <%= session %><br/>
    <%= new Date() %><br/>
    <%ArrayList al = new ArrayList();
    al.add(10);
    %>
    <%= al %>
    <%= new FileOutputStream("ram.txt") %>
    <%! public void service(ServletRequest
req,ServletResponse res)
    throws ServletException,IOException{
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<body bgcolor='yellow'>THIS IS
SERVICE METHOD </body>");
        HttpServletRequest hreq =
(HttpServletRequest)req;
        HttpServletResponse hres =
(HttpServletResponse)res;
        _jspService(hreq,hres);
    } %>
</body>
</html>
```

---



**info will provides a special information about current JSP file. If we want print info attribute information we need use getServletInfo().**

**<%@ page info="this is my special JSP" %>**

**<%=session%><br/>  //valid we didnt write session=false**

**<%="good"%><br/>**

**<%= getServletInfo()%>**

---

```
<html>
<body bgcolor='yellow'>
    <h1>THIS IS TEST2.JSP FILE</h1>
    <%@ page info="this is my test2.jsp file" %>
    <h1><%= getServletInfo() %></h1>
</body>
</html>
```

---

**<%@ page buffer='32kb'  %>**

**<%!**

**%>**

**<%="good"%><br/>**

**----------**

**by default response object always committed on browser.**

**the reason we have attribute like autoFlush='true'**

**once we mention autoFlush='false' response object not commited automatically we need to call out.flush()**

```
<html>
<body bgcolor='yellow'>
    <h1>THIS IS TEST2.JSP FILE</h1>
    <img src=C:\Users\Ramchandar\Desktop\\Koala.jpg
/>
</body>
</html>
```

```
<%@ page autoFlush='false' buffer='32kb' %>
<%
out.println("<html><body>
<img src=C:\\Users\\lenovo\\Desktop\\Koala.jpg />
</body>
</html>");
    out.flush();
%>
```

```
<html>
<body bgcolor='yellow'>
<%@ page isThreadSafe="false" %>
    <%!StringBuilder sb = new StringBuilder(); %>
    <%sb.append("ram");
    Thread.sleep(10000);
    %>
    name: <%= sb %>
```

```
</body>
</html>
```

---

```
<html>
<body bgcolor='yellow'>
<%@ page contentType="application/msword" %>
<% out.print("this msword result"); %>

</body>
</html>
```

---

**Working with exceptions in jsp:**

```
<html>
<body bgcolor='yellow'>
     <% try{
          int a = 10/0;
     }catch(Exception e){
          out.println("dont enter zero as denominator");
     } %>
</body>
</html>
```

---

```
<html>
<body bgcolor='yellow'>
     <%= 10/0 %>
</body>
</html>
```

If we executes above program servlet container will give default error message on browser, if we want to provide user friendly message on browser then we need to go errorPage and isErrorPage

Error1.jsp:

```
<html>
<body bgcolor='yellow'>
```

```
    <%@ page errorPage="error2.jsp" %>
    <%= 10/0 %>
</body>
</html>
```

Error2.jsp

```
<html>
<body bgcolor='yellow'>
    <h1>DONT ENTER ZERO AS DENOMINATOR</h1>
</body>
</html>
```

We have error in the error1.jsp file so automatically controle goes to error2.jsp with the help of errorPage attribute.

If there is no error or exception in our .jsp file there is no usage of errorPage attribute.

How to handle multiple exceptions:

Error1.jsp:

```
<html>
<body bgcolor='yellow'>
<%@ page isErrorPage="true" errorPage="error2.jsp" %>
    <% try{
        int a = 10/0;

    }catch(ArithmeticException e){
        out.println("dont enter zero as denominator");
        throw e;
    }
    catch(NegativeArraySizeException e){
        out.println("declare array size");
    }
    catch(ArrayIndexOutOfBoundsException e){
        out.println("there is no sufficient memeory
location");
```

```
    }
    %>
</body>
</html>
```

Error2.jsp:

```
<html>
<body bgcolor='yellow'>
    <h1>DONT ENTER ZERO AS
DENOMINATOR.....................</h1>
</body>
</html>
```

**If exception is not avialable in try block out put will print on the browser by using out implicit object.**

**If exception is raised in try block we can hold with catch block.**

**With the help of throw keyword and errorPage we parward the controle to another file.**

**If we want hold exception is which is raised in .jsp file we can hold in another like bellow.**

```
<body bgcolor='yellow'>
    <h1>DONT ENTER ZERO AS DENOMINATOR</h1>
    <%@ page isErrorPage="true" %>
    <%=exception %>
</body>
```

**HTML and JSP communication:**

**<body bgcolor='yellow'>**

**<form action='./11.jsp'>**

```
        FirstNumber: <input type='text' name='fno'/><br/>
        SecondNumber: <input type='text'
name='sno'/><br/>
                        <input type='submit' value='submit'/>

</form>
</body>

<%
        int x =
Integer.parseInt(request.getParameter("fno"));
        int y =
Integer.parseInt(request.getParameter("sno"));

%>
Addition is: <%= (x+y) %>

<form action='./r1.jsp'>
        <input type='text' name='name'/><br/>
        <input type='submit' value='submit'/>
```

**</form>**

**<%= request.getParameter("name") %>**

**in the above program there is one drawback that is enduser click on backbutton everytime.**

**add html code directly in JSP and callto JSP directly.**

```
<form action='./r1.jsp'>
    <input type='text' name='name'/><br/>
    <input type='submit' value='submit'/>

</form>

<%= request.getParameter("name") %>
```

**include directive tag:**

**<%@ include file="resource"%>**

**The above tag is used for including the other web resources like .html, .jsp, .text file but not Servlet at translation time statically.**

**Static inclusion means other web resource source code will be copy and paste into source jsp at translation time.**

**For both main jsp and include jsp container will create only one Page Implementation Class.**

**If we using @include directive tag we will includes only content not response, for example if one jsp file(source jsp) communicating with three other jsp files, then the other three jsp files content will be added source jsp file, jsp container will create only one .java file and .class file.**

**Test.jsp:**

```
<html>
    <body bgcolor='yellow'>
        <form>
            <h1>THIS IS TEST.JSP FILE STARTING
PLACE</h1>
            <%@ include file="One.jsp" %>
            <%@ include file="Two.jsp" %>
```

```
            <h1>THIS IS TEST.JSP FILE ENDING
PLACE</h1>
        </form>
    </body>
</html>
```

One.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <H1>THIS IS ONE.JSP FILE</H1>
        </form>
    </body>
</html>
```

Two.jsp:

```
<html>
    <body bgcolor='gold'>
        <form><H1>THIS IS TWO.JSP FILE</H1>

        </form>
    </body>
</html>
```

Three.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <H1>THIS IS THREE.JSP FILE</H1>

        </form>
    </body>
</html>
```

**t1.jsp**

**--------**

**<% out.println("t1 page starting"); %><br/>**

**<%@ include file='t2.jsp' %>**

**<% out.println("t1 page ending"); %><br/>**

**t2.jsp:**

**<%out.println("this t2.jsp page"); %><br/>**

**This tag is used for reuse the content of web resources in different places of project/webapplication. This is used for splitting java code into different pages.**

---

**Test.jsp:**

```
<html>
    <body bgcolor='yellow'>
        <form>
        <h1>THIS IS TEST.JSP FILE STARTING PLACE</h1>
            <%@ include file="One.jsp" %>
            <%@ include file="Two.jsp" %>
        <h1>THIS IS TEST.JSP FILE ENDING PLACE</h1>
        </form>
    </body>
</html>
```

Two.jsp

```
<html>
    <body bgcolor='gold'>
        <form><H1>THIS IS TWO.JSP FILE</H1>
            <%! int a = 555; %>
        </form>
    </body>
</html>
```

One.jsp:

```
<html>
    <body bgcolor='gold'>
        <form>
            <H1>THIS IS ONE.JSP FILE</H1>
            <% int a = 444; %>
        </form>
    </body>
</html>
```

**t1.jsp**

**--------**

```
<% out.println("t1 page starting"); %><br/>

<%@ include file='t2.jsp' %>

<%@ include file='t3.jsp' %>

<%@ include file='t4.jsp' %>

<%= x+y %>

<% out.println("t1 page ending"); %><br/>
```

**t2.jsp**

```
<%out.println("this t2.jsp page"); %><br/>
```

**t3.jsp**

**<% int x = 10; %>**

**t4.jsp**

**<% int y = 10; %>**

**We must be take both t3.jsp and t4.jsp file variable names different otherwise we will get duplicate variable error.**

**In the above program container will create only one Page Implementation Class for all JSP's.**

**If we change any content in one including file, then remaining including files content will be compiled and reloaded that is effected to performance issues.**

**Test.jsp:**

```
<html>
    <body bgcolor='yellow'>
        <form>
        <h1>THIS IS TEST.JSP FILE STARTING PLACE</h1>
            <%@ include file="Four.html" %>
        <h1>THIS IS TEST.JSP FILE ENDING PLACE</h1>
        </form>
    </body>
</html>
```

**Four.html:**

```
<html>
    <body bgcolor='gold'>
        <form>
            <H1>THIS IS FOUR.HTML FILE</H1>
            <%
                int x = 5;
                int y = 15;
            %>
        <h1>The Addtion Is: <%= (x+y) %>  </h1>
        </form>
    </body>
</html>
```

In the above .html file we feel the bellow code normal text. But once this code copied into Test.jsp by @include tag this code will be treated as jsp code. That's why we will output like "The Addition Is: 20".

**Standard actions tags:**

**These are used for includes web resources like Servlet, jsp, html files. All standard actions tag including web resources dynamically.**

**Dynamic inclusion means code is not copy from one web resource to another web resource, only response will be copy.**

**Like in core java method calling statement, whenever we calling method that called method code is not copy only response or result will be copy.**

**The result can't be reusable only printing the content.**

**We have the following action tags:**

**<jsp:include />**

**<jsp:forward />**

**<jsp:params />**

**<jsp:param />**

**<jsp:useBean />**

**<jsp:getBean />**

**<jsp:setProperty />**

**<jsp:getProperty />**

**<jsp:plugin/>**

**<jsp:fallback />**

**<jsp:include>**

**It will used for dynamic inclusion. Code is not copy, only result will be copying. It is equal to rd.include() in servlet. For each including web resource jsp container will create a separate page implementation class.**

**t1.jsp**

```
<% out.println("t1 page starting"); %><br/>
<jsp:include page="t2.jsp"></JSP:include>
<jsp:include page="t3.jsp"></JSP:include>
<jsp:include page="t4.jsp"></JSP:include>
<jsp:include page="test.html"></JSP:include>
<% out.println("t1 page ending"); %><br/>
```

**test.html**

```
<form >
      <input type='text' name='name'/><br/>
      <input type='submit' value='submit'/>
</form>
```

**t2.jsp**

```
from t2.jsp <br/>
```

**t3.jsp**

```
from t3.jsp <br/>
```

**t4.jsp**

```
from t4.jsp <br/>
```

```
<html>
	<body bgcolor="yellow">
		<h1>THIS IS DEMO.JSP FILE STARTS </h1>
		<jsp:include page="R1.jsp"></jsp:include>
		<jsp:include page="R2.jsp"></jsp:include>
		<% out.println("The Result is: "+(x+y)); %>
		<H1>THIS IS DEMO.JSP FILE ENDS</H1>
	</body>

</html>
```

R1.jsp

```
<html>
	<body bgcolor="yellow">
		<h1>THIS IS R1.JSP FILE RESPONSE </h1>
		<%int x=100; %>
	</body>

</html>
```

R2.jsp:

```
<html>
	<body bgcolor="yellow">
		<h1>THIS IS R2.JSP FILE RESPONSE</h1>
		<% int y=200; %>
	</body>

</html>
```

**In the above program we will get error(Demo.jsp).**

**If we are using standard include tag we can not copy the vairables from .jsp to another jsp.**

**html file inclusion:**

**t1.jsp:**

<% out.println("t1 page starting"); %><br/>

<jsp:include page="test.html"></JSP:include>

<% out.println("t1 page ending"); %><br/>

test.html

<%

int x = 5;

int y = 6;

%>

addition: <%= x+y %>

the above html code directly added to browser as a output

---

if we want to do addition operation in html code we need to copy that code

into _JSPService(-,-) with the help <%@ include %>.

t1.jsp:

<% out.println("t1 page starting"); %><br/>

<%@ include file="test.html" %>

<% out.println("t1 page ending"); %><br/>

**test.html**

```
<%
	int x = 5;
	int y = 6;
%>
addition: <%= x+y %>
```

**the above html code will be adding to _JSPService() body.**

**JSP and servlet communication:**

**t1.jsp**

```
<% out.println("t1 page starting"); %><br/>
<jsp:include page='/add' />
<% out.println("t1 page ending"); %><br/>
```

**AddServlet.java:**

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

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class AddServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest
req,HttpServletResponse res)
    throws ServletException, IOException{
        int x =10;
        int y = 20;
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("x+y: "+(x+y));
    }
}
```

**web.xml:**

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

```
        <servlet-name>add</servlet-name>
        <servlet-class>AddServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>add</servlet-name>
        <url-pattern>/add</url-pattern>
    </servlet-mapping>
</web-app>
```

**we can read including JSP file name at run time**

**test.html:**

```
<form action='./t1.jsp'>
    <input type='text' name='name'/>
    <input type='submit' value='submit'/>
</form>
```

**t1.jsp:**

```
<% out.println("t1 page starting"); %><br/>
<jsp:include page='<%=request.getParameter("name")
%>' />
<% out.println("t1 page ending"); %><br/>
```

**t2.jsp:**

```
from t2.jsp <br/>
```

**web.xml:**

```
<web-app>
    <servlet>
        <servlet-name>add</servlet-name>
        <jsp-file>/t1.jsp</JSP-file>
    </servlet>
    <servlet-mapping>
        <servlet-name>add</servlet-name>
        <url-pattern>/t1.jsp</url-pattern>
    </servlet-mapping>
</web-app>
```

| Directive include tag | Standard action include tag |
|---|---|
| --> syntax: <%@ include file="filename" %> | <jsp:include page="filename"/> |
| --> static inclustion | dynamic inclustion |
| --> content copy and paste | only response addes |
| --> only .java file created for including jsp's | based on jsp files multiple .java files and .class files will be create |
| --> content added at transalation time | response added at execution time |
| --> we can use other jsp file variables | we can't use other jsp file variable |
| --> response can reuse | response can be print |
| --> if .jsp file code modifier, other jsp files code also compiler again | if one jsp file code modifier compiler will recompile that .jsp file code only not other jsp file code |
| --> low performance | high performace |
| --> we can't send destination file name dynamically | we can send destination file name dynamically by using expression tag |
| --> we can't communicating with .java file | we can communicate with .java file |

**Test2.html:**

```
<html>
<body bgcolor='RED'>
     <h1>THIS IS Test.html FILE</h1>
     <form action='./t1.jsp'>
          Enter first number: <input type='text' name='fno'/>
          Enter second number: <input type='text'
name='sno'/>
          <input type='submit' value='add'/>
     </form>

</body>
</html>
```

t1.jsp:

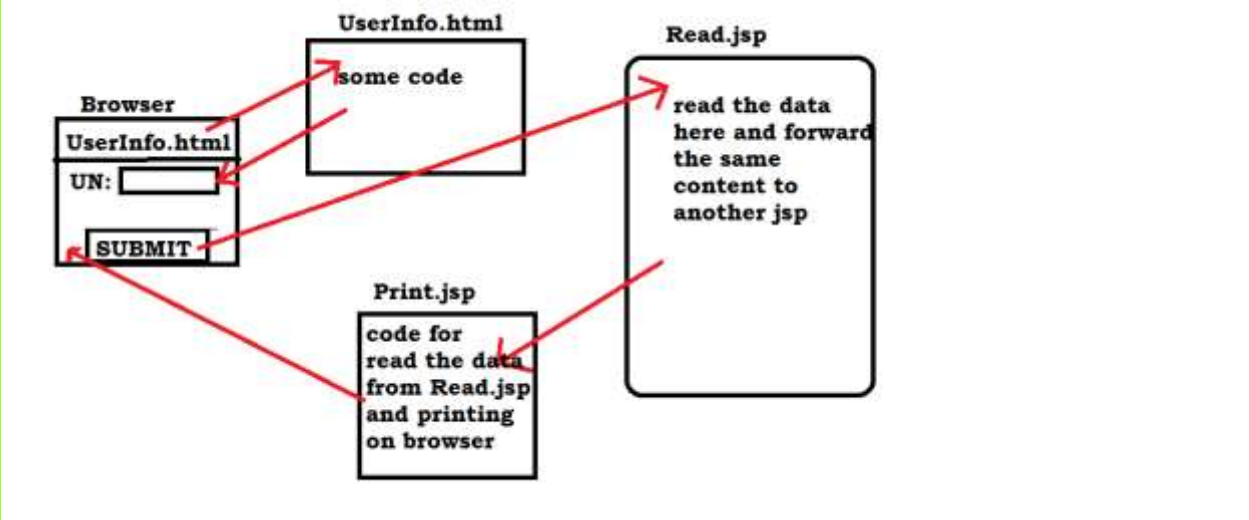
```
<%
     int i = Integer.parseInt(request.getParameter("fno"));
     int j = Integer.parseInt(request.getParameter("sno"));
     int result = i+j;
%>
<jsp:forward page="t2.jsp">
```

```
    <jsp:param value="<%=result %>" name="res"/>
</jsp:forward>
```

## T2.jsp:

```
<html>
    <body bgcolor="yellow">
        <h1>THIS IS R2.JSP FILE RESPONSE</h1>
        <%= request.getParameter("res") %>
    </body>

</html>
```

---




## UserInfo.html:

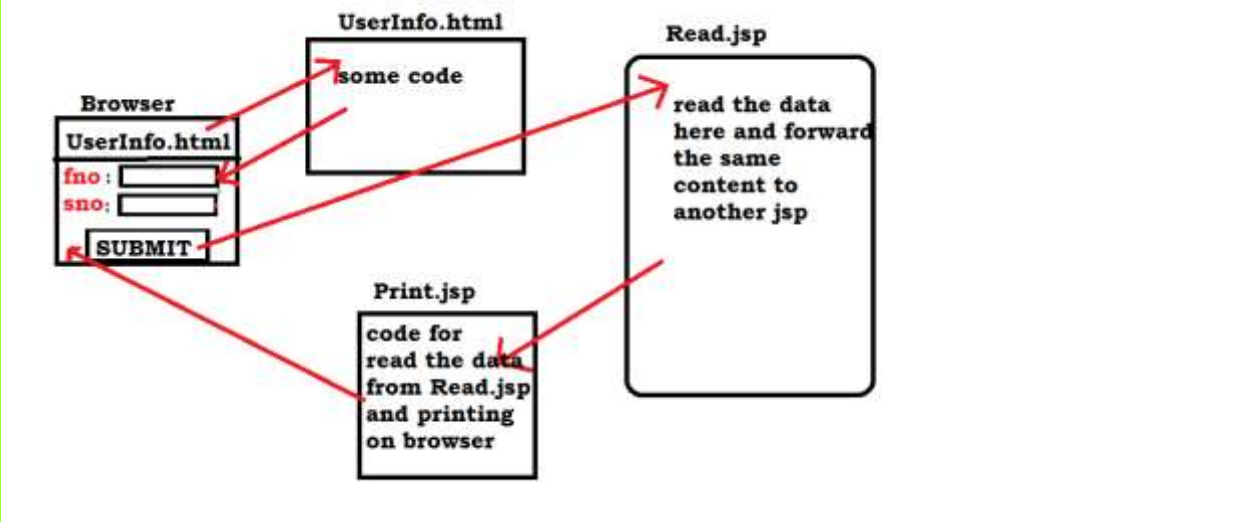
```
<html>
    <body bgcolor='gold'>
        <form action='./Read.jsp'>
    Enter UserName: <input type='text' name='uname'/>
            <input type='submit' value='SUBMIT'/>
        </form>
    </body>
</html>
```

**Read.jsp:**

```
<html>
    <body bgcolor='gold'>
        <form>
<%String username = request.getParameter("uname");%>
<jsp:forward page="Print.jsp">
    <jsp:param value="<%= username %>" name="uname"/>
</jsp:forward>
        </form>
    </body>
</html>
```

**Print.jsp:**

```
<html>
    <body bgcolor='yellow'>
        <form>
<%String username = request.getParameter("uname"); %>
        <%= username %>
        </form>
    </body>
</html>
```




**Add.html:**

```
<html>
    <body bgcolor='gold'>
        <form action='./Read.jsp'>
Enter First  Number:<input type='text'
ame='fno'/><br/>
        Enter Second Number: <input type='text'
name='sno'/><br/>
        <input type='submit' value='ADD'/>
        </form>
    </body>
</html>
```

**Read.jsp:**

```
<html>
    <body bgcolor='gold'>
        <form>
        <%String fno = request.getParameter("fno");
        String sno = request.getParameter("sno");
            int i= Integer.parseInt(fno);
            int j= Integer.parseInt(sno);
            int result=i+j;
```

```
        %>
        <jsp:forward page="Print.jsp">
    <jsp:param value="<%= result %>" name="result"/>
        </jsp:forward>
        </form>
    </body>
</html>
```

**Print.jsp:**

```
<html>
    <body bgcolor='yellow'>
        <form>
<%int result=
Integer.parseInt(request.getParameter("result"));%>
            The Result Is: <%= result %>
        </form>
    </body>
</html>
```

**Another way of writing above program:**

**Add.html:**

```
<html>
    <body bgcolor='gold'>
        <form action='./Read.jsp'>
Enter First  Number:<input type='text'
ame='fno'/><br/>
        Enter Second Number: <input type='text'
name='sno'/><br/>
        <input type='submit' value='ADD'/>
        </form>
    </body>
</html>
```

```
<html>
    <body bgcolor='gold'>
        <form>
        <%String fno = request.getParameter("fno");
        String sno = request.getParameter("sno");
            %>
            <jsp:forward page="Print.jsp">
        <jsp:param value="<%=fno %>" name="fno"/>
        <jsp:param value="<%=sno %>" name="sno"/>
            </jsp:forward>
        </form>
    </body>
</html>
```

**Print.jsp:**

```
<html>
    <body bgcolor='yellow'>
        <form>
            <%int i =
Integer.parseInt(request.getParameter("fno"));
            int j =
Integer.parseInt(request.getParameter("sno"));%>
            The Result Is: <%= (i+j) %>
        </form>
    </body>
</html>
```

Program on <jsp:useBean>,<jsp:setProperty> and <jsp:getProperty>

---

**jsp:useBean tag is provides information to jsp container to create object for particular class**

**It having followings attributes**

**Id → pointing to referenced name**

**Class → pointing to our bean class name**

**Scope → pointing to any one of the scope object.**

**Jsp: setProperty is useful for placing data into our bean variables by calling setter methods.**

**It having following attributes**

**Property → talks about variables name**

**Name → talks about referenced variable name**

**Value → talks about value to variable.**

**Jsp: getProperty: is useful for reading the data from our bean variables by calling getter methods.**

**It having following attributes**

**Property → talks about variables name**

**Name → talks about referenced variable name.**

```
package com.ram;


public class MyBean {

    private String name;

    private String age;

    public String getAge() {
```

```
        return age;
    }

    public void setAge(String age) {
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

test.html:

```
<form action='./t1.jsp'>
    <input type='text' name='name'/>
```

```
        <input type='text' name='age'/>
        <input type='submit' value='submit'/>
</form>
```

**t1.jsp:**

```
--------
<jsp:useBean id="my"
class="com.ram.MyBean"></jsp:useBean>
<%-- <jsp:setProperty property="*" name="my"/> --%>
<jsp:setProperty property="name" name="my"/>
<jsp:setProperty property="age" name="my"/>
<jsp:getProperty property="name" name="my"/>
<jsp:getProperty property="age" name="my"/>
```

```
<web-app>
    <servlet>
        <servlet-name>add</servlet-name>
        <jsp-file>/t1.jsp</JSP-file>
    </servlet>
    <servlet-mapping>
        <servlet-name>add</servlet-name>
```

```
        <url-pattern>/t1.jsp</url-pattern>
    </servlet-mapping>
</web-app>
```

==========

**li.jsp**

```
<jsp:forward page="date.jsp" />
```

**date.jsp**

```
<%= new java.util.Date()
%>
```

==========

**l1.jsp:**

```
<jsp:forward page="date.jsp" >
<jsp:param value="ramchandra" name="name"/>
</JSP:forward>
```

**date.jsp:**

```
<%= new java.util.Date()
%>
<%= request.getParameter("name")
%>
```

---

**<jsp:useBean>**

**11.jsp:**

```
<jsp:useBean id="obj" class="com.ram.Calculation"/>
<%
	int x = obj.cal(5);
	out.print(x);
%>
```

**Calculation.java**

```
package com.ram;

public class Calculation {
	public int cal(int x){
		return x*x*x;
	}
}
```

---

**Custom action tags:**

**Action is a request processing time or runtime instruction to container.**

**We have two types of action tags.**

a. **Standard action tags.**
b. **Custom action tags.**

## Similarities between standard and custom action tags:

a. **Both are action tags.**
b. **Both are runtime instruction to container.**
c. **Both are similar to syntax.**
d. **Both are useful for same purpose, use tags in the foreground and eliminate java code in the foreground, but in the background we are using java code only.**
e. **These tags are using xml syntax but very case sensitive.**

## Differences between standard and custom action tags:

### Standard action tags:

**Predefine tags are always comes along with container, meaning and functionalities already known by the container. Prefix is "jsp". These tags have limited functionalities.**

### Custom action tags:

**These are User define tags. We need to develop by our own. We need to explain meaning and**

**functionalities to container. We can take any name as prefix. Extensible functionalities**

**tag: it is instruction to JSP engine.**

**<prefix:tagname attribute1=value1 ...>**

**tld: tag library descriptor:**

**in this we need to specify the tag and its funcationalities**

**taghandler:**

**it is a java class it will provides the funcationalities of tag**

**taglibrary:**

**collection of tld and taghandler**

**Tag(javax.servlet.jsp.tagext.Tag)**

**setPageContext(pageContext)**

**int doStartTag()**

**int doEndTag()**

**release()**

**For all these methods empty implementation is given by the TagSupport class. This class we can call as adaptor class.**

**We are always define doStatTag() and release() method in our class. Whenever container encounter our tag doStartTag() will be called in this method we are writing our tag functionalities just before our TagHandler object is garbage release() method calls for release the resources.**

**OurTagHandler extends TagSupport implements Tag**

**date.jsp:**

```
<html>
<body bgcolor='red'>
    <%@ taglib prefix="nit" uri="customtags" %>
    <h1>Today date is: <nit:date/> </h1>
</body>
```

**web.xml:**

```
<web-app>
    <jsp-config>
```

```
            <taglib>
                <taglib-uri>customtags</taglib-uri>
                <taglib-location>/WEB-
INF/ourtaglib.tld</taglib-location>
            </taglib>
        </jsp-config>
</web-app>
</html>
```

**ourtaglib.tld**

```
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.2</JSP-version>
    <short-name>nit</short-name>
    <tag>
        <name>date</name>
        <tag-class>com.nit.DateHandler</tag-class>
    </tag>
</taglib>
```

**DateHandler:**

```
package com.nit;

import java.io.IOException;
import java.text.DateFormat;
import java.util.Date;

import javax.servlet.jsp.jspException;
import javax.servlet.jsp.jspWriter;
import javax.servlet.jsp.tagext.TagSupport;

public class DateHandler extends TagSupport{
	public int doStartTag() throws JSPException{
		Date today = new Date();
		DateFormat dft =
DateFormat.getDateInstance(DateFormat.LONG);
		String date = dft.format(today);
		try{
			JspWriter out = pageContext.getOut();
			out.println(date);
		}catch(IOException e){
			System.out.println(e);
```

```
            }
            return SKIP_BODY;
        }
}
```

**Whenever we send request to date.jsp file container will do the**

**following steps.**

1. **container will read userdefine tag (like date)**
2. **container will read userdefine tag "prefix" ( like nit)**
3. **container will matched with taglib prefix**
4. **if both are matched container will read "uri" (like customtags)**
5. **this uri is matched with web.xml file**

    **<taglib-uri> of <taglib>of<jsp-config>**

6. **if both are matched container will <taglib-location>information for .tld file (like ourtaglib.tld) then controle goes to .tld file location.**
7. **container will read about userdefine tag and its taghandler class information in <tag>**

**8. finally controle goes our taghandler class and doStartTag.**

**Program on custom action tags with attributes:**

**greet.jsp:**

```
<html>
    <body bgcolor='wheat'>
        <%@ taglib prefix='nit' uri='customtags' %>
        <nit:greet name="ram" age="35"/>
    </body>
</html>
```

**web.xml:**

```
<web-app>
    <jsp-config>
        <taglib>
            <taglib-uri>customtags</taglib-uri>
            <taglib-location>/WEB-
INF/ourtaglib.tld</taglib-location>
        </taglib>
    </jsp-config>
</web-app>
```

**ourtaglib.tld :**

```
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.2</jsp-version>
    <short-name>nit</short-name>
    <tag>
        <name>greet</name>
```

```
        <tag-class>com.nit.GreetingHandler</tag-
class>
        <body-content>empty</body-content>
        <attribute>
            <name>name</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
        <attribute>
            <name>age</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```

**GreetingHandler.java:**

```
package com.nit;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

public class GreetingHandler extends TagSupport{
    private String name;
    private int age;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
```

```
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public int doStartTag()throws JspException{
        try{
            JspWriter out = pageContext.getOut();
            out.println("hello mr: "+name+"<br/>your
age is: "+age);
        }
        catch(Exception e){
            System.out.println("--------------");
            e.printStackTrace();
        }
        return SKIP_BODY;
    }
}
```

**Write program on custom action tags with body:**

**hello.jsp:**

```
<html>
  <body>
   <%@ taglib prefix = "ex" uri = "customtags"%>
     <ex:Hello>
        This is my own tag body content
     </ex:Hello>
   </body>
</html>
```

web.xml:

```
<web-app>
    <jsp-config>
        <taglib>
            <taglib-uri>customtags</taglib-uri>
            <taglib-location>/WEB-
INF/ourtaglib.tld</taglib-location>
        </taglib>
    </jsp-config>
</web-app>
```

**ourtaglib.tld:**

```
<taglib>
   <tlib-version>1.0</tlib-version>
   <jsp-version>1.2</jsp-version>
   <short-name>ex</short-name>

   <tag>
      <name>Hello</name>
      <tag-class>com.nit.HelloTag</tag-class>
      <body-content>scriptless</body-content>
   </tag>
</taglib>
```

**HelloTag.java:**

```
package com.nit;

import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;

public class HelloTag extends SimpleTagSupport {
```

```
    StringWriter sw = new StringWriter();
    @Override
    public void doTag()throws JspException,
IOException {
        getJspBody().invoke(sw);

getJspContext().getOut().println(sw.toString());
    }
}
```

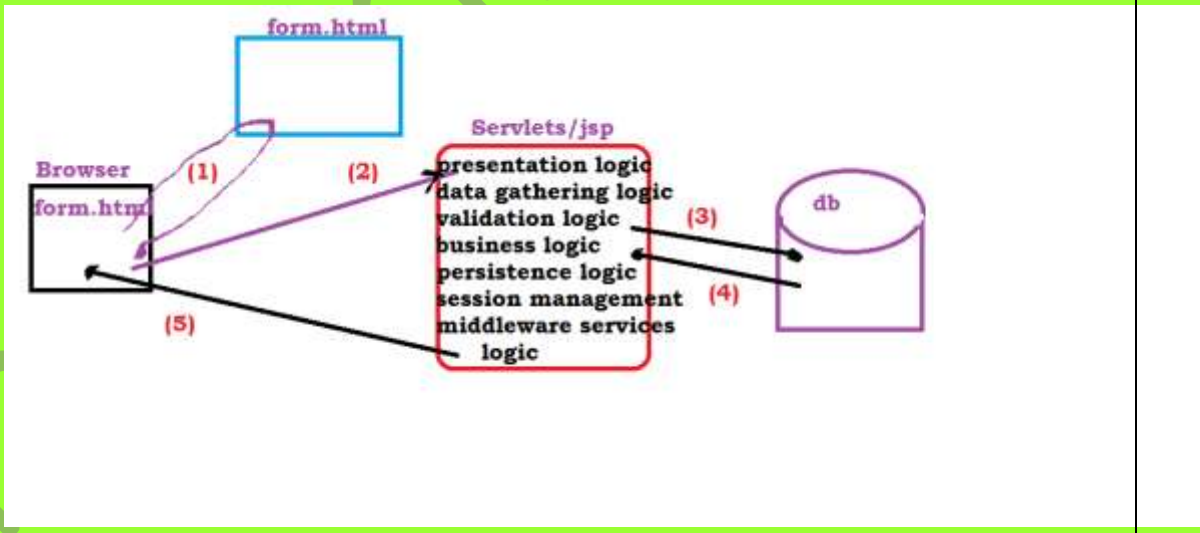
---

## Web Module Architectures:

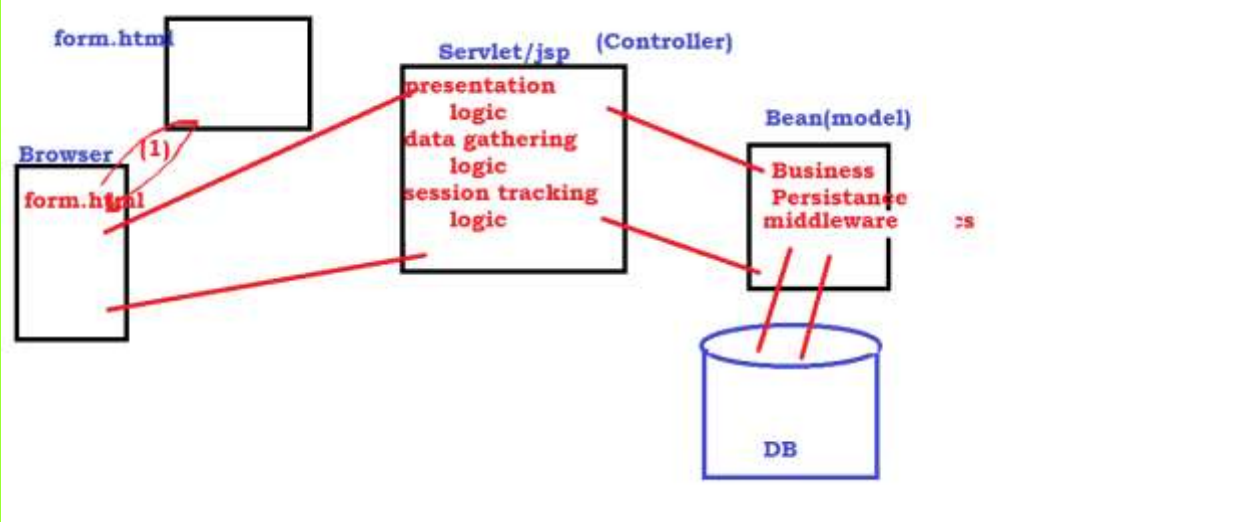
---

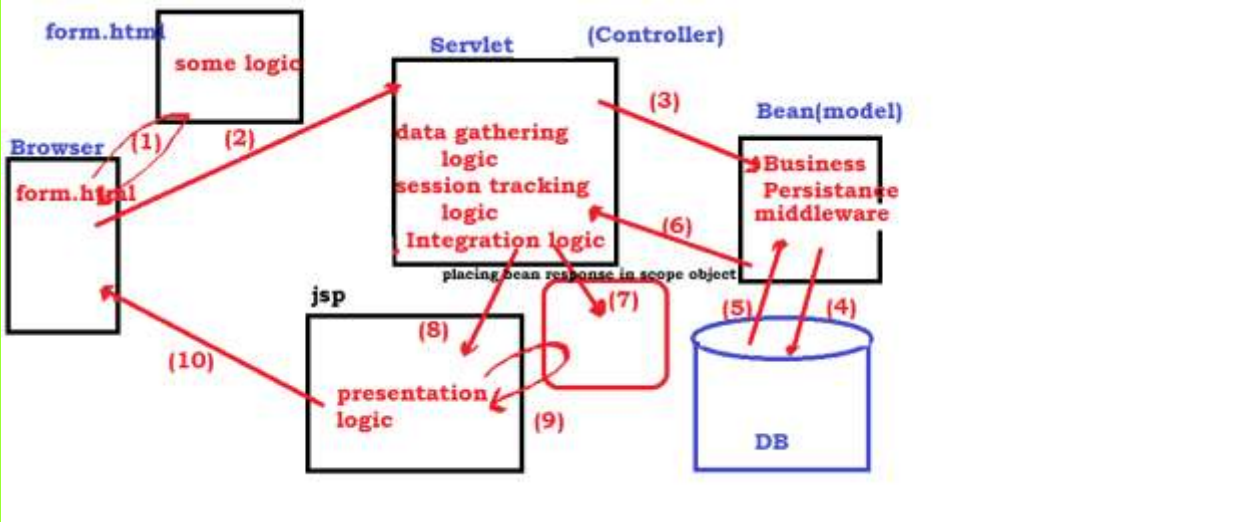
### Two models

#### a. Model-1




#### b. Model-2




**mvc1:**

**emp.html**

```
<html>
<body bgcolor="yellow">
    <form action='./mvc'>
    eno:<input type='text' name="eid"/>
        <input type='submit' value='submit'/>
```

```
        </form>

</body>
</html>
```

**web.xml:**

```
<web-app>
        <servlet>
                <servlet-name>mvc</servlet-name>
                <servlet-class>ControleServlet</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>mvc</servlet-name>
                <url-pattern>/mvc</url-pattern>
        </servlet-mapping>

</web-app>
```

**ControleServlet**

```
import java.io.IOException;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.nit.Employee;

public class ControleServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest
req,HttpServletResponse res)throws
ServletException,IOException{
        int eid =
Integer.parseInt(req.getParameter("eid"));
        Employee ebean = new Employee();
        ebean.setEid(eid);
        getServletContext().setAttribute("ebean",
ebean);
```

```
        getServletContext().getRequestDispatcher("/view.jsp").forward(req, res);

                System.out.println("from doGet");
        }

}
```

**Employee:**

```
package com.nit;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class Employee {
        private int eid;
        private String ename;
```

```
    private int esal;
    Connection con;
    public Employee(){
        try{

    Class.forName("oracle.jdbc.driver.OracleDriver");
            con =
DriverManager.getConnection("jdbc:oracle:thin:@localh
ost:1521:xe","system","manager");
        }catch(Exception e){
            System.out.println("problem in empliyee
class");
            e.printStackTrace();
        }
    }
    public int getEid() {
        return eid;
    }
    public void setEid(int eid) {
        this.eid = eid;
        result();
    }
```

```
    public String getEname() {
        return ename;
    }
    public void setEname(String ename) {
        this.ename = ename;
    }
    public int getEsal() {
        return esal;
    }
    public void setEsal(int esal) {
        this.esal = esal;
    }
    public void result(){
        Statement st;
        try {
            System.out.println("con"+con);
            st = con.createStatement();
            ResultSet rs = st.executeQuery("select *
from employee");
            rs.next();
            ename=rs.getString(2);
```

```
                esal = rs.getInt(3);
                System.out.println("ename: "+ename);
                System.out.println("esal: "+esal);
            } catch (SQLException e) {
                System.out.println("problem in result");
                e.printStackTrace();
            }


    }
}
```

**view.jsp**

```
<html>
<body bgcolor='green'>
<%@ page import="com.nit.Employee"
scope=”application”%>
<jsp:useBean id="ebean" class="com.nit.Employee"
scope="application"></JSP:useBean>
eno: <jsp:getProperty property="eid" name="ebean"/>
```

**ename: <jsp:getProperty property="ename" name="ebean"/>**

**esal: <jsp:getProperty property="esal" name="ebean"/>**

**<%= "controle comes here" %>**

**</body>**

**</html>**

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

**mvc:**

**login.html:**

```
<body bgcolor='yellow'>
    <form action='./login' method='post'>
        <input type='text' name='name'/>
        <input type='password' name='password'/>
        <input type='submit' value='submit'/>
    </form>

</body>
```

**web.xml:**

```
<web-app>
    <servlet>
        <servlet-name>logic</servlet-name>
        <servlet-class>LoginServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>logic</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>
</web-app>
```

**LoginServlet.java**

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

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.nit.LoginBean;


public class LoginServlet extends HttpServlet{

      public void doPost(HttpServletRequest
req,HttpServletResponse res)throws
ServletException,IOException{

            res.setContentType("text/html");

            PrintWriter out = res.getWriter();

            String name = req.getParameter("name");

            String password =
req.getParameter("password");

            LoginBean bean = new LoginBean();

            bean.setName(name);

            bean.setPassword(password);


            req.setAttribute("bean", bean);

            boolean flag = bean.validate(password);


            if(flag){
```

```
                RequestDispatcher rd =
req.getRequestDispatcher("./sucess.jsp");
                rd.forward(req, res);
            }
            else
            {
                RequestDispatcher rd =
req.getRequestDispatcher("./failure.jsp");
                rd.forward(req, res);
            }

        }

}
```

**LoginBean.java**

```
package com.nit;


public class LoginBean {
    String name;
    String password;
```

```
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public boolean validate(String password){
        if(password.equals("ram")){
            return true;
        }
        else
            return false;
    }
}
```

**failure.jsp:**

**this is failure JSP**

**sucess.jsp**

**<%@ page import="com.nit.LoginBean" %>**

**this sucess JSP**

**<% LoginBean bean = (LoginBean) request.getAttribute("bean");**

**%>**

**<%= bean.getName()**

**%>**

**In java we can develop webapplication in the following two models**

1. **Model1**
2. **Model2 (MVC)**

**Model1:**

**In this approach we are using either Servlet or jsp as web component. In this model we are always writing the logic within the either one Servlet or one JSP.**

**Disadvantages:**

1. **No clean separation between logics.**
2. **Modification done at one logic remaining logics will be effected.**
3. **Maintenance and enhancement is very complex.**
4. **Parallel development is not possible**
5. **Productivity will be decreases.**
6. **We must be aware about all technologies.**

**It is not industry recommended approach**

**Advantages:**

**No parallel development no needs to use more programmers. (Less human resources).**

**Model2:**

**We have types of architectures.**

**MVC-1:**

**Controller: The component which is behavior like taking the request, validate the request, and handover to destination resource and gathering the response and finally handover to browser is called controller.**

**Model:**

**This is one java object, which is useful for interacting with database and gathering the response and handover to controller.**

**View:**

**It is useful for holding logic related view or representation.**

**Disadvantages:**

1. **No Clear separation between logics**
2. **If we are doing modifications on one logic some of the remaining logics also affected.**
3. **The problem here either Servlet or jsp will takes all request and responses.**

**Advantages:**

**Compare model-1 this approach is some provides modularity.**

**The drawbacks which we have in mvc-1 and model-1 we can avoid all these things through MVC-2 architecture.**



**Expression Language:**

**It is simplifies the reading the data from bean object and other implicit objects.**

**And also provides arithmetic,logical,relational operations.**

**How to evaluates the expression in el:**

```
<html>
<head>
 <title>Expression language example1</title>
</head>
<body>
${1<2}
${1+2+3}
</body>
</html>
```

**output**

**true 6**

**----------**

**Test.jsp**

```
<html>
<head>
```

```
 <title>Expression language example2</title>
</head>
<body>
<form action="display.jsp">
Student Name: <input type="text" name="stuname"
/><br>
Student RollNum:<input type="text" name="rollno"
/><br>
<input type="submit" value="Submit Details!!"/>
</form>
</body>
</html>
```

**display.jsp**

```
<html>
<head>
<title>Display Page</title>
</head>
<body>
 Student name is ${ param.stuname } <br>
 Student Roll No is ${ param.rollno }
```

```
</body>
</html>
```

---

**reading the data from application scope:**

**Test.jsp**

```
<html>
 <head>
 <title>EL example3</title>
 </head>
 <body>
 <%
 application.setAttribute("author", "ramchandra");
 application.setAttribute("Site", "Nareshit.com");
 %>
 <a href="display.jsp">Click</a>
 </body>
 </html>
```

**display.jsp**

```
<html>
```

```
<head>
<title>Display Page</title>
</head>
<body>
${applicationScope.author}<br>
${applicationScope.Site}
</body>
</html>
```

---

```
<html>
<head>
<title>EL example3</title>
</head>
<body>
<%
session.setAttribute("author", "ramchandra1");
session.setAttribute("Site", "Nareshit1.com");
%>
<a href="display.jsp">Click</a>
</body>
```

```
</html>


<html>
 <head>
 <title>Display Page</title>
 </head>
 <body>
 ${sessionScope.author}<br>
 ${sessionScope.Site}
 </body>
 </html>
-----------------------------------
<html>
 <head>
 <title>EL example3</title>
 </head>
 <body>
 <%
 request.setAttribute("author", "ramchandra2");
 request.setAttribute("Site", "Nareshit2.com");
```

```
%>
 ${requestScope.author}<br>
${requestScope.Site}
<a href="display.jsp">Click</a>

</body>
</html>

<html>
<head>
<title>Display Page</title>
</head>
<body>
${requestScope.author}<br>
${requestScope.Site}
</body>
</html>

---------------------------------------------
<html>
```

```
<head>
<title>EL example3</title>
</head>
<body>
<%
request.setAttribute("author", "ramchandra2");
request.setAttribute("Site", "Nareshit2.com");
pageContext.forward("/display.jsp");
%>
 </body>
</html>
```

**display.jsp:**

```
<%= request.getAttribute("author") %>
author: ${requestScope.author}
```

---

**JSTL**

**Java Server Pages Standard Tag Library**

**Fast Development**

**Code reusebility**

**No need to Scriptlet tag in JSP file**

**It will provies following tags like**

- **core tags**
- **functional tags**
- **sql tags**
- **formating tags**
- **xml tags**

**Core Tags:**

**<%@ taglib uri="http://java.sun.com/JSP/jstl/core" prefix="c" %>**

**c:out:**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
```

```
<title>Tag Example</title>
</head>
<body>
<c:out value="${'Advanced Java'} "/>
</body>
</html>
```

**c:out  like include**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Tag Example</title>
</head>
<body>
<c:import var="data" url="hi.txt"/>
<c:out value="${data}"/>
</body>
</html>
```

**c:set**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Tag Example</title>
</head>
<body>
<c:set var="x" scope="session" value="${1000*5}"/>
<c:out value="${x}"/>
</body>
</html>
```

**c:remove**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
```

```
<c:set var="money" scope="session"
value="${4000*4}"/>

<p>Before Remove Value is: <c:out
value="${money}"/></p>

<c:remove var="money"/>

<p>After Remove Value is: <c:out
value="${money}"/></p>

</body>

</html>
```

**c:catch**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>

<html>

<head>

<title>Core Tag Example</title>

</head>

<body>


<c:catch var ="e">

   <% int x = 2/0;%>

</c:catch>
```

```
<c:if test = "${e != null}">
   <p>The type of exception is : ${e} <br />
   There is an exception: ${e.message}</p>
</c:if>

</body>
</html>
```

c:if

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session"
value="${4000*4}"/>
<c:if test="${income > 8000}" >
```

```
   <p>My income is: <c:out value="${income}"/><p>
</c:if>
</body>
</html>
```

```
<c:choose> <c:when> <c:otherwise>
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="income" scope="session"
value="${4000*4}"/>
<p>Your income is : <c:out value="${income}"/></p>
<c:choose>
    <c:when test="${income <= 1000}">
       good salary
    </c:when>
    <c:when test="${income > 10000}">
```

```
        very good salary
    </c:when>
    <c:otherwise>
       Income is undetermined...
    </c:otherwise>
</c:choose>
</body>
</html>
```

**c:forEach**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:forEach var="j" begin="1" end="3">
   count <c:out value="${j}"/><p>
</c:forEach>
```

```
</body>

</html>
```

:c:forTokens:

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>

<html>

<head>

<title>Core Tag Example</title>

</head>

<body>

<c:forTokens items="ram chandra rao" delims=" "
var="name">

   <c:out value="${name}"/><p>

</c:forTokens>

</body>

</html>
```

c:url

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
```

```
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:url value="/index1.jsp" var="url">
 <c:param name="a" value="10"/>
 <c:param name="b" value="20"/>
</c:url>
${url}
</body>
</html>
```

c:redirect

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
```

```
<body>
  <c:set var="url" value="0" scope="request"/>
  <c:if test="${url<1}">
     <c:redirect url="http://google.com"/>
  </c:if>
  <c:if test="${url>1}">
     <c:redirect url="http://facebook.com"/>
  </c:if>
</body>
</html>
```

**Functional Tags:**

**contains:**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>
<html>
```

```
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="String" value="Welcome to nareshit"/>

<c:if test="${fn:contains(String, 'nareshit')}">
   <p>Found nareshit string<p>
</c:if>

<c:if test="${fn:contains(String, 'NARESHIT')}">
   <p>Found NARESHIT string<p>
</c:if>

</body>
</html>
```



**containsIgnoreCase**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
```

```
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>


<c:set var="String" value="Welcome to nareshit"/>


<c:if test="${fn:containsIgnoreCase(String, 'nareshit')}">

   <p>Found nareshit string<p>

</c:if>


<c:if test="${fn:containsIgnoreCase(String,
'NARESHIT')}">

   <p>Found NARESHIT string<p>

</c:if>


</body>
```

```
</html>


endsWith

<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>

<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>


<c:set var="String" value="Welcome to nareshit"/>


<c:if test="${fn:endsWith(String, 'nareshit')}">

  <p>yes<p>

</c:if>


<c:if test="${fn:endsWith(String, 'NARESHIT')}">
```

```
   <p>NO<p>
</c:if>


</body>
</html>
```

**indexOf:**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>
<html>
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="string1" value="It is first String."/>
<c:set var="string2" value="It is this second String."/>
```

```
<p>Index-1 : ${fn:indexOf(string1, "first")}</p>
<p>Index-2 : ${fn:indexOf(string2, "second")}</p>

</body>
</html>
```

**trim:**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>
<html>
<head>
<title>Using JSTL Functions</title>
</head>
<body>

<c:set var="str1" value="Welcome to JSP
programming        "/>
```

```
<p>String-1 Length is : ${fn:length(str1)}</p>


<c:set var="str2" value="${fn:trim(str1)}" />
<p>String-2 Length is : ${fn:length(str2)}</p>
<p>Final value of string is : ${str2}</p>


</body>
</html>
```

**startsWith**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>
<html>
<head>
<title>Using JSTL Function</title>
</head>
<body>
<c:set var="msg" value="The Example of JSTL
fn:startsWith() Function"/>
```

```
The string starts with "The": ${fn:startsWith(msg, 'The')}

<br>The string starts with "Example":
${fn:startsWith(msg, 'Example')}

</body>

</html>



split and join

<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>

<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>

<html>

<head>

<title>Using JSTL Functions</title>

</head>

<body>


<c:set var="str1" value="Welcome-to-JSP-
Programming."/>

<c:set var="str2" value="${fn:split(str1, '-')}" />
```

```
<c:set var="str3" value="${fn:join(str2, ' ')}" />
<p>String-1 : ${str1}</p>
 <p>String-2 : ${str2}</p>
<p>String-3 : ${str3}</p>
<c:set var="str4" value="${fn:split(str3, ' ')}" />
<c:set var="str5" value="${fn:join(str4, '-')}" />

<p>String-5 : ${str5}</p>

</body>
</html>
```

**toLowerCase:**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>
<html>
<head>
```

```
<title> Using JSTL Function </title>
</head>
<body>
<c:set var="string" value="WELCOME TO JSP
PROGRAMMING"/>
${fn:toLowerCase("HELLO,")}
${fn:toLowerCase(string)}
</body>
</html>
```

**toUpperCase:**

```
<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>
<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>
<html>
<head>
<title> Using JSTL Function </title>
</head>
<body>
```

```
<c:set var="string" value="Welcome to JSP
Programming"/>

${fn:toLowerCase("hello,")}

${fn:toUpperCase(string)}

</body>

</html>
```

```
othermethods:

<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>

<%@ taglib
uri="http://java.sun.com/JSP/jstl/functions"
prefix="fn" %>

<html>

<head>

<title> Using JSTL Function </title>

</head>

<body>

<c:set var="string" value="Welcome to JSP
Programming"/>

${fn:toLowerCase("hello,")}  <br/>
```

```
${fn:toUpperCase(string)}  <br/>
${fn:substring(string, 5, 17)}  <br/>
${fn:substringAfter(string, "to")}<br/>
${fn:substringBefore(string, "JSP")}<br/>
${fn:substringBefore(string, "JSP")}<br/>
Length of the String is: ${fn:length(str2)}
${fn:replace(string, "JSP", "JAVA SERVER PAGES")}
</body>
</html>
```

**Formating tags:**

**used formating about message date number**

```
<%@ taglib prefix="c"
uri="http://java.sun.com/JSP/jstl/core"%>
<%@ taglib prefix="fmt"
uri="http://java.sun.com/JSP/jstl/fmt"%>
<html>
<head>
```

```
<title>fmt:formatDate</title>
</head>
<body>
<h2>Different Formats of the Date</h2>
<c:set var="Date" value="<%=new java.util.Date()%>" />
<p>
Formatted Time :
<fmt:formatDate type="time" value="${Date}" />
</p>
<p>
Formatted Date :
<fmt:formatDate type="date" value="${Date}" />
</p>
<p>
Formatted Date and Time :
<fmt:formatDate type="both" value="${Date}" />
</p>
<p>
Formatted Date and Time in short style :
<fmt:formatDate type="both" dateStyle="short"
timeStyle="short"
```

```
value="${Date}" />

</p>

<p>

Formatted Date and Time in medium style :

<fmt:formatDate type="both" dateStyle="medium"
timeStyle="medium"

value="${Date}" />

</p>

<p>

Formatted Date and Time in long style :

<fmt:formatDate type="both" dateStyle="long"
timeStyle="long"

value="${Date}" />

</p>


sqltags:

<%@ taglib uri="http://java.sun.com/JSP/jstl/core"
prefix="c" %>

<%@ taglib uri="http://java.sun.com/JSP/jstl/sql"
prefix="sql"%>

<html>

<head>
```

```
<title>sql:setDataSource Tag</title>
</head>
<body>

<sql:setDataSource var="db"
driver="oracle.jdbc.driver.OracleDriver"
    url="jdbc:oracle:thin:@localhost:1521"
    user="system"  password="manager"/>

<sql:query dataSource="${db}" var="rs">
SELECT eid,ename,esal from employee
</sql:query>

<c:forEach var="table" items="${rs.rows}">
<c:out value="${table.eid}"/>
<c:out value="${table.ename}"/>
<c:out value="${table.esal}"/>


</c:forEach>
<sql:update dataSource="${db}" var="count">
```

```
INSERT INTO employee VALUES (102,'sam',4000)
</sql:update>
</body>
</html>
```