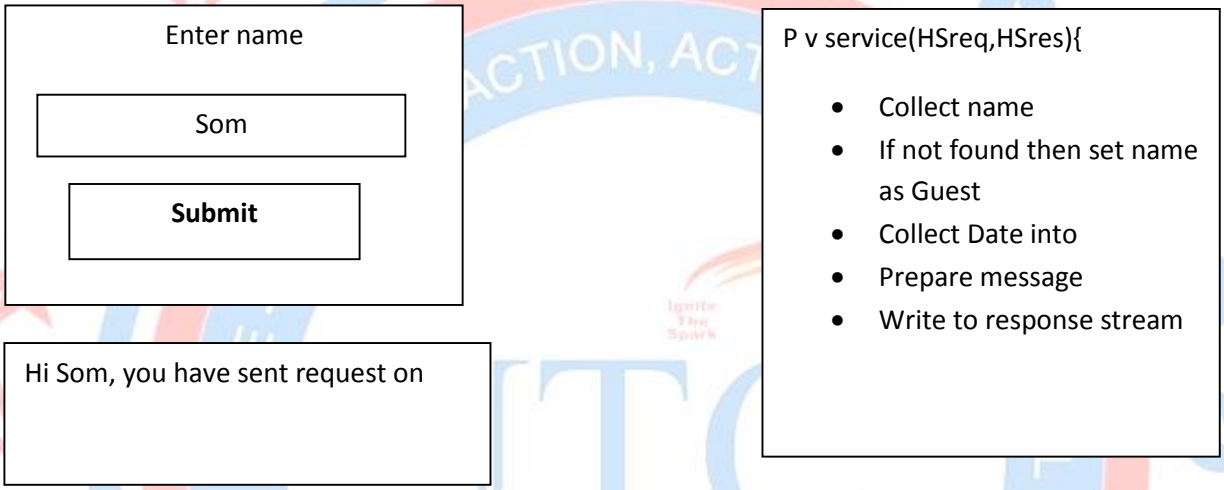
Introduction

- JSP stands for Java Server Pages.
- JSP is a server sid Technology like Servlet which is used to develop server sid web components.
- JSP is designed as presentation Technology which allows you to render the dynamic pesponse easily.
- JSP is combination of HTML and Java Code.
- The extension of the file must be .sp.

Case 1: Servlet for Request procession& Presentation



Jtc1:

Index.html

```
<html>
<body>
<h1>java training Center</h1>
<h3>Account Login</h2>
<form action=”test.jtc”method=”post”>
<input type=”text”name=”fname”/>
<br/>
<input type=”submit”/>
</form>
</body>
</html>
```

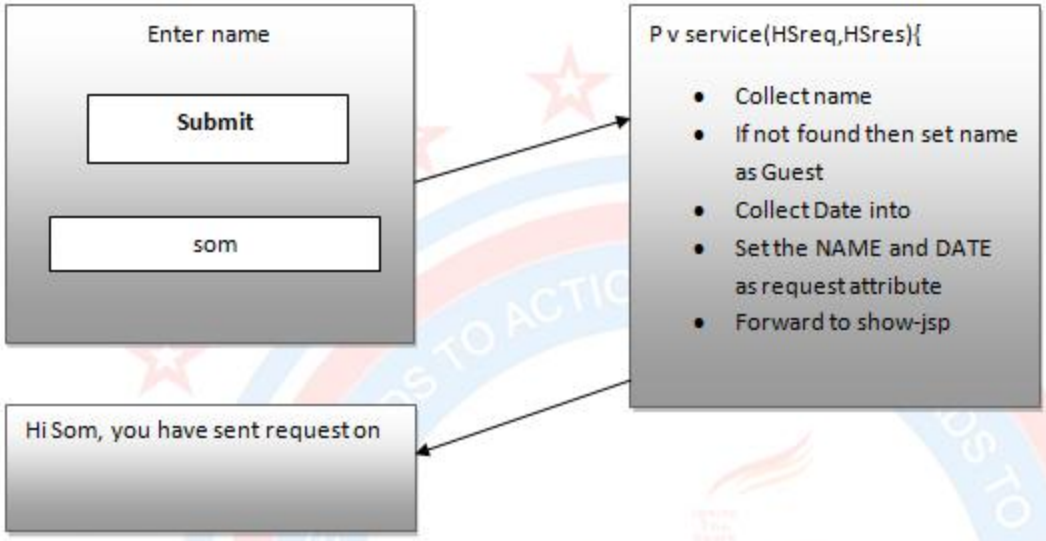
Web.xml

```
<web-app>
<welcome-file-list>
<welcome-file>index.html</welcome.file>
</welcome-file-list>
<Servlet>
<Servlet-name>testsErvlet</Servlet-name>
<Servlet-class>com.jtcindia.servlets.TestServlet
</Servlet-class>
</Servlet>
< Servlet -mapping>
<Servlet-name>test Servlet<.servlet-name>
<url-pattern>/test.jtc,/url-pattern>
</Servlet-mapping>
</web-app>
```

TestServlet.java

```
Package com.jtcindia.servlets;
Import java.io.I;
Import java.util.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
Public class TestSErvlet extends Http serv let{
Puoctected void service (HttpServletRequest,
HttpServletResponse rewponse ) throws
servletException,IO Exception{
String fnm=request.getParameter(“fname”);
Date dt=new Date();
If(fnm ==null|| fnm.trim().length()==0) {
Fnm=”Guest”;
}
String msg=”<h1>Hi +fnm+”<br/> you hava sent
request on “+dt;
Response.setContentTypt(“text/html”);
Response.getWrite().write(msg);
}
}
```

Case 1: Servlet for Request procession& JSP for Presentation



Jtc2:

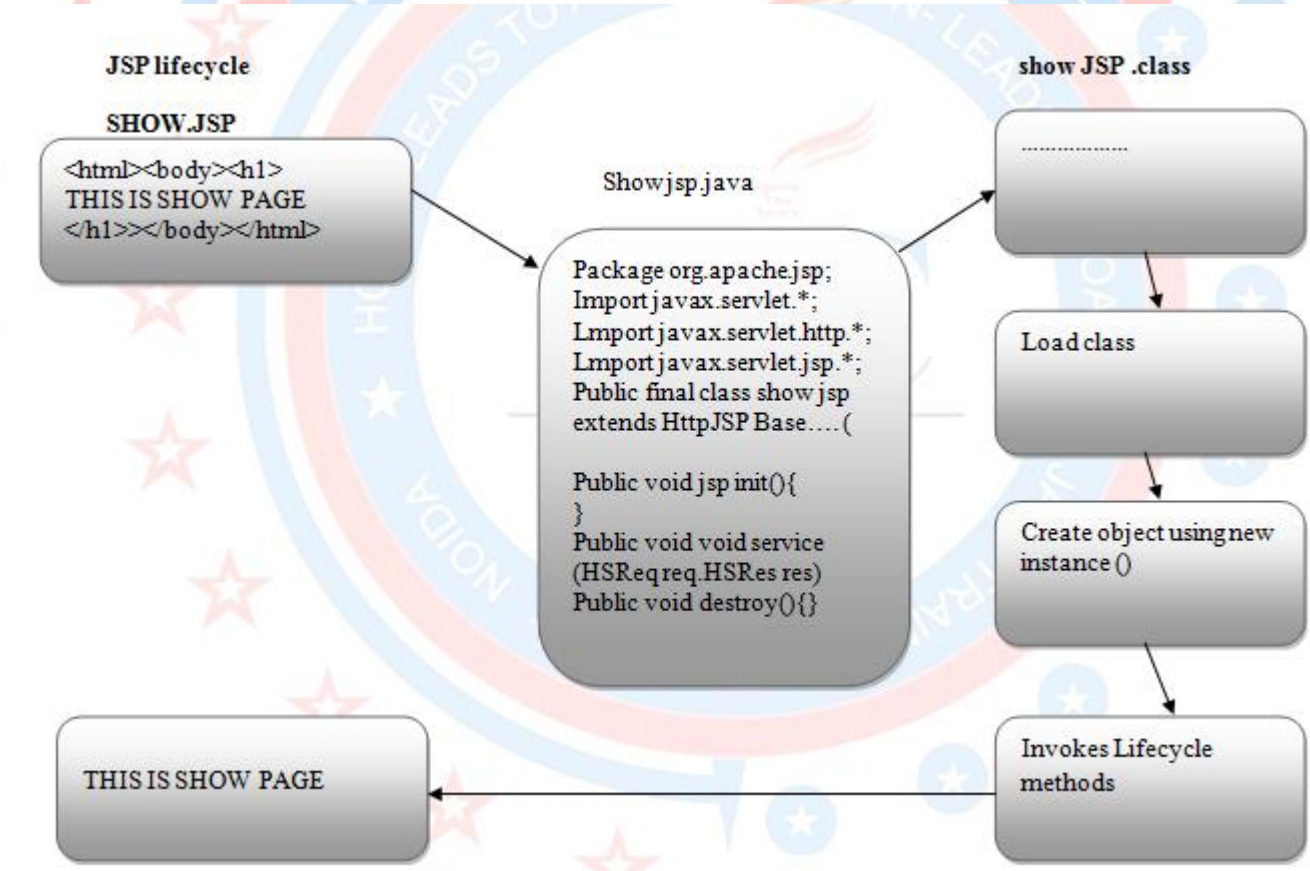
Index.html	Web.xml
<html> <body> java training Center Account Login <form action=”test.jtc”method=”post”> <input type=”text”name=”fname”/> <input type=”submit”/> </form> </body> </html>	<web-app> <welcome-file-list> <welcome-file>index.html</welcome.file> </welcome-file-list> <Servlet> <Servlet-name>testsservlet</Servlet-name> <Servlet-class>com.jtcindia.servlets.TestServlet </Servlet-class> </Servlet> < Servlet -mapping> <Servlet-name>test Servlet<.servlet-name> <url-pattern>/test.jtc,/url-pattern> </Servlet-mapping> </web-app>

Test Servlet.java

```
Package com.jtcindia.servlets;
Import java.io.I;
Import java.util.*;
Import javax.servlet.*;
Import javax.servlet.http.*;
/*
*@Author:som prakash rai
*@company:java Training center
*@visit   :Www.jtcindia.org
**/
Public class TestSErvlet extends Http serv let{
```

```
Puoctected void service (HttpServletRequest,
HttpServletResponse rewponse ) throws
servletException,IO Exception{
String fnm=request.getParameter("fname");
Date dt=new Date();
If(fnm ==null|| fnm.trim().length()==0) {
Fnm="Guest";
Request.setAttribute("NM".fnm);
Request.setAttribute("DT",dt);
Request.get
RequestDispatcher("show.jsp").forward(request,response);
}
}
```

Jsp Lifecycle



- Whenever client sends the first request to JSP the Container will do the following task:
  - Translates JSP to Servlet (show.jsp > show jsp.java)
  - Compiles the Translated Servlet (show jsp.jaba > show jsp.clas)
  - Loads the Translated Servlet class.
  - Creates the instance of translated Servlet class.
  - Calls the lifecycle method jsp lnit() for initialization.
  - Invokes thd lifecycle method jsp service().
- Whenever client sends the 2 owards request to JSP then jsp service () will be called directly.
- At container shutdown time, container calles the lifecycle method jspDestroy();

JSP Life Cycle methods

- Following three are the lify cycle method of JSP;
  - Public void jsplnit()
  - Public void jspService (HttpServletRequest res,HttpServletResponse res)
  - Public void jspDestroy()

Show.jsp

```
<html><body><h1>THIS IS SHOW PAGE</h1></body></html>
```

Steps

- Create Dynamic web project with the name Jtc 3
- Create the JSP file with the name show.jsp
- Open the folder
  <WORKSPACE>\/metadata\/plugins\org.eclipse.wst.server.core\tmp0\work\catalina\localhost\
- Deploy the application to server.
- Start the server
- Send the request to show.jsp
  http://localHost:9090/Jtc3/show.jsp
- Open the folder
  <WORKSPACE>\/metadata\/plugins\org.eclipse.wst.server.core\tmp0\work\catalina\localhost\Jtc3\
  - Org\apache\jsp
  - Show jsp.java
  - Show jsp.class

  .

JSP Implicit Objects

- **JSP** implicit objects are the objects which are readly available in every JSP.
- Every JSP you are writing will be translated Servlet class be the web container.
- Every translated Servlet will have jsp Service() method which contains mainly 9 local
- Variables to hold web container objects.

Public void jsp Service (HttpServletTequest request, HttpServletTesponse response) throws jav a.io.IOException, servletException {

pageContext PageContext=null;
HttpSession Session=null;
Servlet Context application=null;
servletConfig config=null;
jspWriter out=null
objcect page =this;
Throwable exception=null;
……
}

- You can not define these objects your own in JSP page.

JSP Implicit Object type	JSP Implicit object
HttpServletRquest	Request
HttpSErvletResponse	Response
pageContext	dpageContext
HttpSession	Session
servletContext	Application
servletConfig	Config
JspWriter	Out
Object	Page
Throwable	exception

JSP scriptiong Elements

JSP Scripting Elements are used to use java Statements in JSP.

1. Scriptlets
2. Expressions
3. Declarations

Scriptlets

Syntax:
<%
….
%>
	Ex:
	<% int a =33;
	Out.println(a);
	System.out.println(a);
	%)
		jspService(){
		……
		Int a=33;
		Out.println(a);
		System.out.println(a);
		…….
		}

- Any valid java statements are allowed inside the scriptlets.
- All the statements inside the scriptlet will be palced inside the jsp Service() method of translated Servlet.

Expressions:

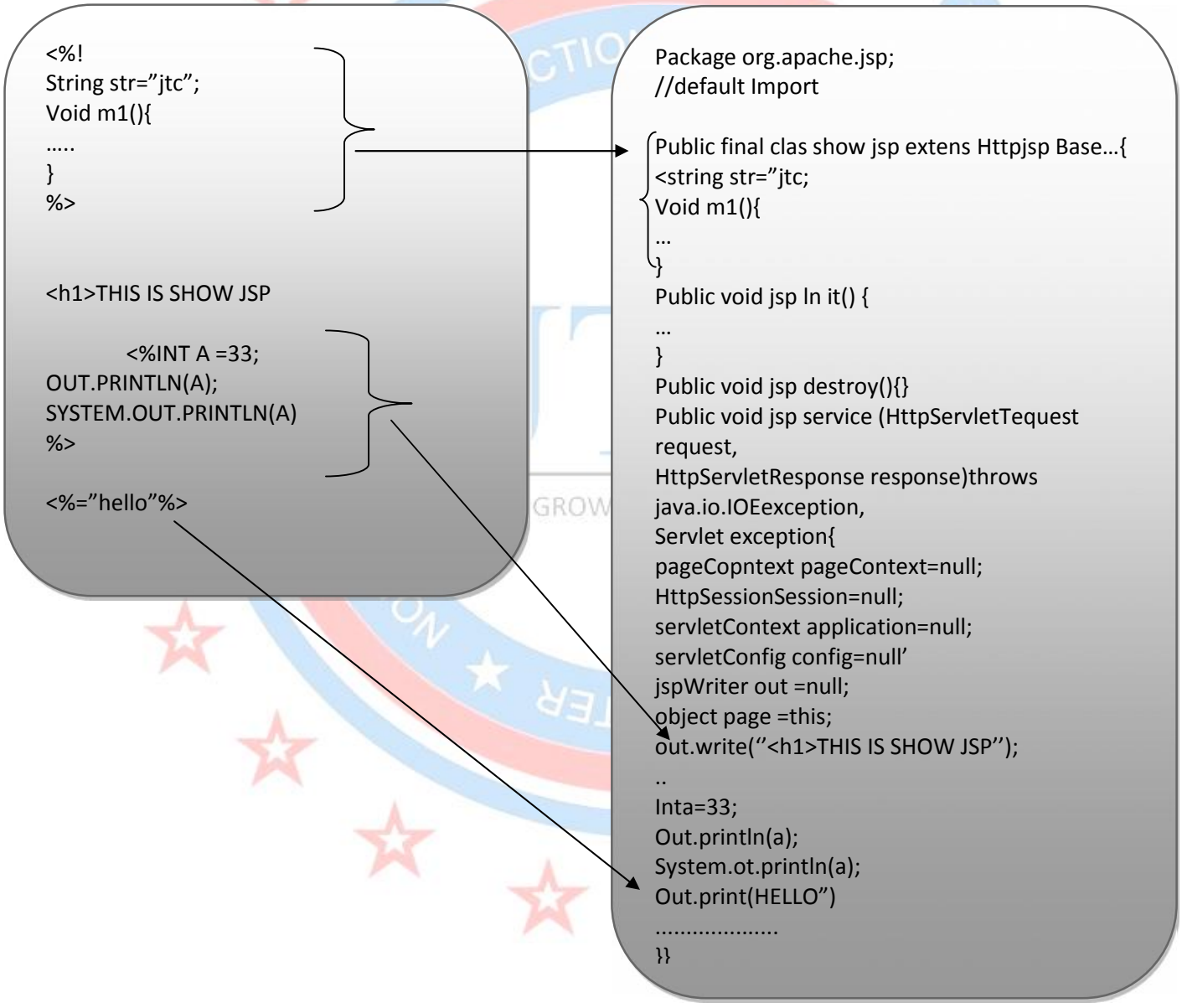
Syntax:
<%=exp
%>
	Ex:
	<% string str=jtc%)
	<%=str%>
	<%=Hello%>
		jspService(){
		……
		String str=”jtc”;
		Out.println(str);
		Out.println(Hello)
		…….
		}

Declarations

Syntax:
<%
….
%>
	Ex:
	<% !
	String str=”jtc”;
	Void m1(){
	}
	%)
		Class show jsp…{
		String str=”jtc”
		Void m1(){
		……
		jspService(){
		…….
		}

- Method definitions, Blocks, constructors, class level variables are allowed inside the Declarations.
- All the code inside the Declarations will be palced directly inside the translated Servlet class and out
- side the jsp service () method.

Jtc 4:



You can use the following to define comments in JSP page

<!—THIS IS HTML COMMENT→
<%--THIS IS JSP COMMENT /hidden Comment--%>

Servlets	Jsp
You have to write the Servlet class be extending Http Servlet or generic Servlet	Container weites or generates the Servlet class by extending HttpjspBase
You have to compile the Servlet class written by you.	Container compiles the Translated sergvlet class.
Servlet lifecycle methods are Public void init (servletConfig) public void service (HSRequest req.HSRes res) Public void destroy()	Jsp lifecycle methods are Public void jsplnit()/jspinit() Public void jsp service (HSRequest.req.HSRes res) Public void jspdestroy()/jspDestroy()
Servlet will be initialized at container start up or at first request.	JSP will be initialized at container start up or at first request.
When you modify the Servlet then you have to recompile, re-deploy and re-start the server.	When you modify thje JSP, container will do the automatic re-translation, re-compilation etc.
You can define any package for Servlet.	For the jsp, Translated Servlet will be placed in org. apache jsp package.

Index.jsp

```
<html><body>
<h1>JAVA Training center</h1>
<%!
Public void jsplnit() {
System.out.println("jsplnit()");
}
Public void ispDestroy() {
System.out.println(" jspDestroy()
}
%>
</body></html>
```

An error occurred at line:30 in the generated java file
Duplicate method jsplnit() in type index jsp
An error occurred at line: 35 in the generated java file
Duplicate methoed jsp Destroy () in type index jsp

Q1) What is JSP ?
Ans: Refer Notes.

Q2) what are the lifecycle methods of JSP?
Ans:

1. Jsplnit()/jsplnit()
2. JspService()
3. jspDestroy()/ jspDestroy()

Q3) How many times jspInit () method will be called?
Ans: Only once.

Q4) How many times jspService()method will be called?
Ans: Every time when client will send request.

Q5) How many times jsp destroy() method will be called?
Ans: Only once.

Q6) How can I write Java Statements in JSP?
Ans: Refer Notes.

Q7) What is difference between Servlet and JSP
Ans: At first request.

Q8) when the JSP will be translated to the Servlet?
Ans: At first request.

Q9) Con I initialize the JSP at container startup?
Ans: Yes

Q10) Can I define the config parameter for the JSP?
Ans: yes

Q11) How can I initialize the JSP at container startup?
Ans:

```
<Servlet>
<Servlet-name>tests</Servlet-name>
<jsp-file>/show.jsp</jsp-file>
<load-on-startup>1</load-on-startup>
</Servlet>
<Servlet-mapping>
<Servlet-name>tests</Servlet-name>
<url-pattern>/show.jsp</url-pattern>
</Servlet-mapping>
```

Q12) How can I define the Config Parameter for the JSP?
Ans:

```
<Servlet>
<Servlet-name>tests</Servlet-name>
<jsp-file>/show.jsp</jsp-file>
```

```
        <init-param>
        <param-name>contact</param-name>
        <param-value>Noida Branch</param-value>
        </init-param>
        </Servlet>
        <Servlet-mapping>
        <Servlet-name>tests</Servlet-name>
        <url-pattern>/show.jsp</url.pattern>
        </Servlet-mapping>
In  JSP:
        <h1>
        THIS IS SHOW JSP
        String con =config.getInit parameter("contact");
        Out.println(con);
        %>
</h1>
```

Q13) where can I find Servlet API related classes?
Ans: TOMCAT HOME/LIB/Servlet-api.jar

Q14) where can I find Servlet API related classes?
Ans: TOMCAT HOME/LIB/jsp-api.jar

Q15) Which web component is faster either serviette or JSP.
Ans: First Request:
Servlet is faster then JSP when you send the first request.
Second Request Onwards:
Both Servlet and JSP takes the same amount of time for processing 2nd Request on wards.

Q16) What is the difference between jsp Init () and jsp init () in jsp?
Ans: In Tomcat 5, when container is writing translated Servlet then it is placing only jsp Service () method inside the translated Servlet. So you can override both jspInit() and jspInit() methods.
Tomcat 5, when container is writing translated Servlet then it is placing jsp Service () and jspInit() methods inside the translated Servlet. So you can override only jsp lnit () and you can not override jsp init () methods.

Q17) What is the difference between jsp Destroy() or jsp Destroy () in jsp?
Ans: in Tomcat 5, when container is writing translated Servlet then it is placing only jsp service () method inside the translated Servlet. So you can override both jspDestroy () and jsp Destroy () methods.

In Tomcat 5, when container is writing translated Servlet then it is placing jsp Service () and jsp Destroy () methods inside the translated Servlet. So you can override only jsp Destroy () and you can not override jsp Destroy () methods.

Q18) Can we use the request, response, out implicit object in the jsp init () or not?
Ans: No Because implicit object will be local variable to jsp service () .

Q19) What are JSP Implicit objects?
Ans: Implicit object is an object which can be used by you directly without creating or getting the object.

There are 9 Implicit objects in jsp

jspWriter	out
HttpServletRequest	request
HttpServletResponse	response
Httpsession	session
ServletContext	application
servletConfig	config
object	page
throwable	exception
pageContext	pageContext

In the above implicit objects following object can be enabled or disabled.

httpSession	session
throwable	exception

By default session will be enabled and exception will be disables.

Q20) What is the package name for translated Servlet?
Ans: org.apach.jsp

Q21) What is the default super class for translated servlets?
Ans:HttpJspBase

JSP Directives

- Directive is a special instruction to container to perform the required task.
- There are 3 directives in jsps

1. Include directive
2. Page directive
3. Taglib directive

1. Include directive:

- Include directive is used to include one jsp or html in another jsp.
- Include directive line will be peplaced with the content of included JSP or HTML at translation time
- You can write one or more include directives inside the jsp.

Syntax:

<%@include file+"some jsp or html"%>

Ex:

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

<%@ include file =”login.jsp”%>
<%@ include file =”footer.html”%>

Header.html

<h2>java Training Center</h2>

Footer.html

<h3>All Rights Reserved.java Training center.2015</h3>

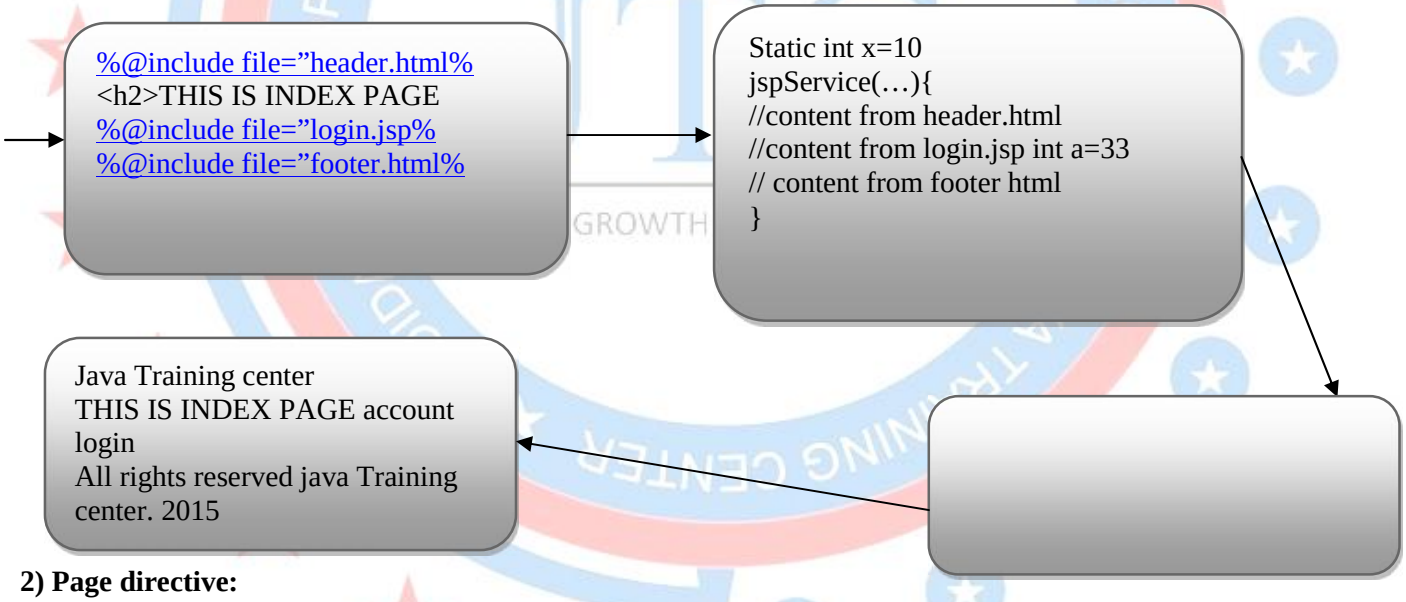
Login.jsp

<h1>Account Login</h1>
<%!STATIC INT X=10;%>
<% int a =10

Index.jsp

%@include file=”header.html%
<h2>THIS IS INDEX PAGE
%@include file=”login.jsp%
%@include file=”footer.html%

Processing steps



2) Page directive:

Syntax:

<%@page
Language=””
Session=””
Extends=
isThreadSafe=””

errorPage=""
isErrorPage=""
isELIgnored=""
%>

Language:

- Language attribute is used to specify the language for the script lets and declarations.
- Currently valid value is java and is the default value.
  <%@page language="java"%> VALID
  <%@page language="L"%> INVALID (INVALID LANGUAGE ATTRIBUTE)

Import:

- Import attribute is used to specify the packages to be imported for the translated Servlet.
- You can specify one or more packages with comma() separation using this attribute.
  %@page language="java.util.*java.io*"%

Session:

- Session attribute is used to enable or disable session object in the JSP.
- By default session object is enabled in JSP.
- If you want to disable use the following.
  <%@page session="false"%>

Index.jsp

```
<%@ page import="java.util.*"session=true"%>
<%@page language="java"%>
<html>
<body>
<h1>
THIS IS JSP<br/>
<%+session%<% NO COMPILATIONERROR
%>
<br>
<%=session.getid()%>
</h1>
</body>
</html>
```

Index.jsp

```
<%@ page import="java.util.*"session=true"%>
<html>
<body>
<h1>
THIS IS JSP<br/>
<%+session%<% COMPILATION ERROR %>
<br>
<%=session.getid()%>
</h1>
</body>
</html>
```

Extends:

- httpJspBase is the default super class for the translated Servlet.
- If you want to use other supen class for the translated Servlet instead of HttpJspBase, use extends attribute.
- When you take Http Servlet as a super class then you have to override any of the methods of HttpServlet.
  <%@page extends="javax.servlet.http.Servlet""%>

Index.Jsp	**Index.Jsp**
%@page import="java.io.IOException"% %@page extends="javax.servlet.http.HttpServlet"% <html> <body> THIS IS JSP </body> </html>	%@page import="java.io.IOException"% %@page extends="javax.servlet.http.HttpServlet"% <html><body> <%! Public void service (HttpServletRequest req, HttpServletResponse res) throws IOException, Servlet Exception{system.out.println ("Http service Method Called"); jspService (req,res); } %> THIS IS JSP </body></html>

Is ThreadSafe:

- This attribute is used to specify the Servlet thread model required.
- Default Servlet Thread model is multi thread model.
- Default value of is ThreadSafe attribute is true.
- If you want to follow single thread model, use this attribute value as false.
  <%@page is ThreadSafe="false"%>
- If you want to follow multi thread model, use this attribute value as false.
- <%@page is ThreadSafe="true"%>

Index.Jsp	**Index.Jsp**
%@page is Threadsafe="true"% <html> <body> THIS IS JSP </body> </html>	%@page is Threadsafe="false"% <html> <body> THIS IS JSP </body> </html>

Error Page and Is Error page:

- If went to handle the exceptions in a centralized JSP then use these two attributes.
- Error page is used to specify the centralized JSP.
- Is Error page is used to enable the exception implicit object in the centralized JSP used for error heading.

Index.jsp	**Index.jsp**
%@page errorPage=”err.jsp”% <html><body> THIS IS JSP <% Int x=10 / 0 %> </body></html>	%@page errorPage=”true”% <html><body> <font color =”red”size=”6”>ERROR OCCURRED <%=exception%> </h1></body></html>

Is ELIgnored:

- Is ELIgnored attribute is used to enable or disable EL expressions.

Index.jsp	Index.jsp
%@page errorPage=”true”% <html><body> THIS IS INDEX PAGE <% Request.setAttribute(“Name”,”som prakash”); %> Name:${NAME} 	%@page errorPage=”false”% <html><body> THIS IS INDEX PAGE <% Request.setAttribute(“Name”,”som prakash”); %> Name:${NAME}

Note:

Upto Tomcat 5.0, EL Expressions are ingnored by default. You have to enable them.
From tomcat 6.0 ,EL Expressions are enabled by default.

3.Taglib directive:

Taglib directive is used to use the custom tags in JSP.

JSP Standard Actions

- Following are the important JSP Standard actions
  - <jsp:include>
  - <jsp:forward>
  - <jsp:param>
  - <jsp:useBean>
  - <jsp:setProperty>
  - <jsp:getProperty>

<jsp:include>

- <jsp:include> is used to include jsp or html another jsp.
- <jsp:include> will include the response of the include page at runtime.

Syntax:

<jsp:include page=”jsp or html file”/>

Ex:

<jsp:include page="header.jsp"/>
<jsp:include page="footer.html"/>

- <jsp:include> functionality is similar to include() method of RequestDispatcher.

```
<%
request.setAttribute("email",som@jtcindia.org);
RequestDispatcher rd=request.getRequestDispatcher("header.jsp");
rd.include(request,response);
%>
```

<jsp:forward>

- <jsp:forward> is used to forward the control from one JSP to another JSP or html

Sysntax:

<jsp:forward page="jsp or html file"/>
<jsp:include page="hello.jsp"/>

- <jsp:forward> functionality is similar to forward() method of RequestDispatcher.

```
<%
        request.setAttribute("email","som@jtcindia.org");
        RequestDispatcher rd=request.getRequestDispatcher("hello.jsp");
        Rd.forward(request,response);

%>
```

<jsp:param>

- <jsp:param> is used to send the parameter data from One to another JSP while you are including or frowarding.
- <jsp:param> must be used along with <jsp:include> or <jsp:forward>

Syntax:

<jsp:param name="email" value="som@jtc.com"/>

Ex:

```
<jsp:Includepage="header.jsp">
<jsp:paramvalue="Som@jtcindia.org"name="email"/>
</Jsp:Include>
<jsp:Forwardpage="home.jsp">
<jsp:paramvalue="Som@jtcindia.org"name="email"/>
</jsp:Forward>
```

Jtc5: Files Required

1.header.jsp	2.footer.jsp	3.home.jsp
4.test.jtc	5.Noida.jsp	6.GrNoida

1.header.jsp
`<html>`
`<head>`
`<metahttp-equiv="Content-Type"content="text/html; charset=ISO-8859-1">`
	3.home.jsp
	`<html>`
	`<head>`
	`<metahttp-equiv="Content-Type"content="text/html;`

```
<title>Insert title here</title>
</head>
<body>
<h1>${param.cname }</h1>
</body>
</html>
```

2.footer.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>All Rights Reserved.${param.cname
}.2012</h1>
</body>
</html>
```

4.test.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
String bn=request.getParameter("brance");
if(bn.equals("Noida")){
%>
<jsp:forwardpage="Noida.jsp">
<jsp:paramvalue="Noida"name="Noida"/>
<jsp:paramvalue="Noida@jtcindia.org"name="emai
l"/>
</jsp:forward>
<%
}elseif(bn.equals("GRN")){
        %>
}
<jsp:forwardpage="GrNoida.jsp">
<jsp:paramvalue="GRN"name="GRN"/>
<jsp:paramvalue="GrNoida@jtcindia.org"name="em
ail"/>
</jsp:forward>
<%
}%>
</body>
</html>
```

```
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:includepage="header.jsp">
<jsp:paramvalue="Java Training
Center"name="cname"/>
</jsp:include>
<formaction="test.jsp">
<h1>Select the Branch</h1>
<selectname="branch">
<optionvalue="Noida">Noida</option>
<optionvalue="GR.Noida">GR.Noida</op
tion>

</select>
<inputtype="submit"value="Submit"/>
</form>
<jsp:includepage="footer.jsp">
<jsp:paramvalue="Java Training
Center"name="cname"/>
</jsp:include>
</body>
</html>
```

5.Noida.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:includepage="header.jsp">
<jsp:paramvalue="Java Training
Center"name="cname"/>
</jsp:include>

This is Noida.jsp Home Page<br>
which shows ${param.bn } branch
details Contact email:${param.email
}

<jsp:includepage="footer.jsp">
<jsp:paramvalue="Java Training
Center"name="cname"/>
</jsp:include>
</body>
</html>
```

6.GrNoida.jsp

```
<html>
```

```
<head>
<metahttp-equiv="Content-
Type"content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:includepage="header.jsp">
<jsp:paramvalue="Java Training
Center"name="cname"/>
</jsp:include>
This is GrNoida.jsp Home Page<br>
which shows ${param.bn } branch
details Contact email:${param.email
}
<jsp:includepage="footer.jsp">
<jsp:paramvalue="Java Training
Center"name="cname"/>
</jsp:include>
</body>
</html>
```

<jsp:useBean>,<jsp:setProperty>,<jsp:getProperty>

- These 3 tags are used for
  - Creating Java object
  - Storing data in java Bean Object
  - Accessing data from Java Bean Object.

<jsp:useBean>

- <jsp:useBean> is used to create the java Bean Object and to store that object in the required scope.

Syntax:

<jsp:useBean id="" class="" scope=""/>

Ex:

<jsp:useBean id="ST" class="com.jtcindia.Student" scope="session">

<jsp:setProperty>

- <jsp:setProperty> is used to set the value to the property of java bean Object.

Syntax:

<jsp:setProperty name="<id/attName>" property="<nameOfVariable>" value="……"/>

Ex:

<jsp:setProperty name="ST" property="name" value="som"/>

<jsp:getProperty>

- getProperty tag is used to get the value of the property of javabean object.

Syntax:

<jsp:getProperty name="<id/attName>" property="<nameofVariable>"/>

Ex:

<jsp:getProperty name="ST" property="name"/>

Jtc6:

1.test1.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>
<%
Student st=new Student();
st.setSid(99);
st.setName("som");
st.setPhone(9876543);
session.setAttribute("STUD",st);

%>
<%
Student
std=(Student)session.getAttribute("
STUD");
%>
Sid:<%=std.getSid() %><br/>
Name:<%=std.getName() %><br/>
phone:<%=std.getPhone() %><br>

</h1>
</body>
</html>
```

2.test2.jsp

```
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

<hr/>USING useBean and other tags
<jsp:useBeanid="ST"class="com.jtcindia.Student"scope=
"session"/>
<jsp:setPropertyproperty="sid"name="ST"value="88"/>
<jsp:setPropertyproperty="name"name="ST"value="Prakas
h"/>
<jsp:setPropertyproperty="phone"name="ST"value="98765
34"/>
<br/>
Sid:<jsp:getPropertyproperty="sid"name="ST"/><br/>
Name:<jsp:getPropertyproperty="name"name="ST"/><br/>
Phone:<jsp:getPropertyproperty="phone"name="ST"/>
</body>
</html>
```

3.Student.java

```
package com.jtcindia;

publicclass Student {
	privateintsid;
	private String name;
	privatelongphone;
//Setters & getters
}
```

- By using the <jsp:setProperty> you can store the data in the java Bean object by collecting the data from request parameters.

Jtc7:

1.index.html

```
<!DOCTYPEhtml>
<html>
<head>
<metacharset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<formaction="result.jsp"method="pos
```

2.show.jsp

```
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<jsp:useBeanid="ST1"class="com.bean.Jtc4"></jsp:useB
```

```
t">
<h1>
<br/>Enter Sid<br/>
<inputtype="text"name="sid">
<br/>Enter Name<br/>
<inputtype="text"name="Name">
<br/>Enetr Phone<br/>
<inputtype="text"name="phone">
<br>Enetr Another phone<br/>
<inputtype="text"name="studphone">
</h1>
<inputtype="submit"value="SHOW"/>
</form>

</body>
</html>
```

```
ean>
<jsp:setPropertyproperty="sid"name="ST1"/>
<hr/><hr/>
<h1>

Sid:<jsp:getPropertyproperty="sid"name="ST1"/>
Name:<jsp:getPropertyproperty="name"name="ST1"/>
Phone:<jsp:getPropertyproperty="phone"name="ST1"/>
<br/><br/>
</h1>
<br/>

</body>
</html>
3.student.java
package com.bean;

publicclass Jtc4 {
	privateintsid;
	private String name;
	privatelongphone;
//Setters & Getters
```

Include directory	Include action
Index.jsp <%@include file="com.jsp"%>	<%@include page="com.jsp"%>
Atrribute of include directive is file <%@include file="header.jsp"%>	Atrribute of include directive is file <%@include file="header.jsp"%>
Include directive line will be replaced with content of included JSP or HTML at transaction time.	Include action line will be replaced with response of JSP or HTML include at runtime.
Include directive is not good in the performannce.	Include you are including any jsp using action, you cann pass parameters to that JSP using <jsp:param>.

Q22) What is the use of pageContext implicit object.

Ans: pageContext is the master implicit object which allows to access other implicit objects of JSP. You can manage the scopes of JSP using this object.

```
<html>
<body>
<%
pageContext.setAttribute("MSG","Page Scope Message");
pageContext.setAttribute("MSG","Request Message",PageContext.REQUEST_SCOPE);
```

```
pageContext.setAttribute("MSG","Session Message",PageContext.SESSION_SCOPE);
pageContext.setAttribute("MSG","Context Message",PageContext.APPLICATION_SCOPE);
pageContext.setAttribute("MSG","Error Message",pageContext.APPLICATION_SCOPE);
%>
<h1>Using WebbComponenet Object<br/>
<%
String msg1=request.getAttribute("MSG").toString();
String msg2=request.getAttribute("MSG").toString();
String msg3=request.getAttribute("MSG").toString();
%>
<%=msg1 %><br/>
<%=msg2 %><br/>
<%=msg3 %><br/>
Using Page Context Object<br/>
<%
String data1=pageContext.getAttribute("MSG").toString();
String data2=pageContext.getAttribute("MSG",PageContext.PAGE_SCOPE).toString();
String data3=pageContext.getAttribute("MSG",pageContext.REQUEST_SCOPE).toString();
String data4=pageContext.getAttribute("MSG",pageContext.SESSION_SCOPE).toString();
String data5=pageContext.getAttribute("MSG",pageContext.APPLICATION_SCOPE).toString();
%>
<%=data1 %><br/>
<%=data2 %><br/>
<%=data3 %><br/>
<%=data4 %><br/>
<%=data5 %><br/>

<%
Object obj1=pageContext.findAttribute("MSG");
Object obj2=pageContext.findAttribute("ERROR");
%>
<%=obj1 %><br/>
<%=obj2 %><br/>
</h1>
</body>
</html>
```

Scopes

- You can store the data in different objects to access later depending on the requirement.
- These objects are called as scopes.
- You can store the data in the required scope as an attribute.
- An attribute is name - value pair.
- Name of an attribute is of type String.
- Value of an attribute is of type Object.
- Attributes are read-write i.e You have to store the attribues so that you can collect later.

You can use the following methods to manage the attributes

1. void setAttribute(String,Object)
2. Object getAttribute(String)
3. void removeAttribute(String)

4. Enumeration getAttributeNames()

Servlet Scopes

There are 3 scopes in Servlet.

1) Request Scope
2) Session Scope
3) Context Scope

- Request Scope data can be accessed by single user within the same request before delivering the response.
- Session Scope data can be accessed by single user across multiple requests.
- Context Scope data can be accessed by multiple users across multiple requests.

JSP Scopes

There are 4 scopes in JSP.

1. Page Scope
2. Request Scope
3. Session Scope
4. Application Scope

- Page Scope data can be accessed by within the same jsp page(transalated servlet)
- Request Scope data can be accessed by single user within the same request before delivering the response.
- Session Scope data can be accessed by single user accross multiple requests.
- Application Scope data can be accessed by multiple users accross multiple requests.
- You can store attributes directly in request, session and application but You can not store attributes directly in page because it is of type java.lang.Object.

JSP EL Expressions

SYNATX:
${OneObject.AnotherObject.PropertyName}

Following are various EL implicit objects.

1. param
2. paramValues
3. header
4. headerValues
5. cookie
6. pageScope
7. requestScope
8. sessionScope
9. applicationScope

10. initParam
11. pageContext

******ELJtc1 (WB): Explanation******

param and paramValues

- These two EL implicit objects are used to collect the request parameters.
- When use param then internally request.getParameter() method will be called.
- When use paramValues then internally request.getParameterValues() method will be called.

header and headerValues

- These two EL implicit objects are used to collect the request Headers.
- When use header then internally request.getHeader() method will be called.
- When use headerValues then internally request.getHeaderValues() method will be called.

cookie

- This EL implicit object is used to collect the request

InitParam

- This EL implicit object is used to collect the context parameters from ServletContext object.

  Usage:

  ${initParam.state}
  ${initParam.city}

pageContext :

- This EL implicit object allows to access all other JSP implicit objects and their properties.

Usage:

${pageContext.session.id}
${pageContext.request.remoteAddr}
${pageContext.request.method}
${pageContext.request.requestURI}
${pageContext.response.contentType}

Note: copy jstl.jar and standard.jar files to WEB-INF/lib folder from E:\Tomcat7.0\webapps\jsp-examples\WEB-INF\lib

In JSP, You can store and access the attributes using master implicit object called pageContext.

You can use the following methods of PageContext to manage the attributes.

1. void setAttribute(String,Object)
2. void setAttribute(String,Object,int)
3. Object getAttribute(String)
4. Object getAttribute(String,int)
5. void removeAttribute(String)
6. void removeAttribute(String,int)
7. Enumeration getAttributeNamesInScope(int)
8. Object findAttribute(String)

```
session.setAttribute("email","som@jtc");
pageContext.setAttribute("email","som@jtc",PageContext.SESSION_SCOPE);

request.setAttribute("email","som@jtc");
pageContext.setAttribute("email","som@jtc",PageContext.REQUEST_SCOPE);

pageContext.setAttribute("email","som@jtc");
pageContext.setAttribute("email","som@jtc",PageContext.PAGE_SCOPE);
```

Jtc8: files required

1.index.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Java Training Center</h1>
<formaction="test.jtc"method="post">
<table>
<tr>
<td>Student Name</td>
<td><inputtype="text"name="sname"/></td></tr>
<tr>
<td>Primary Email</td>
<td><inputtype="text"name="email"/></td></tr>

<tr>
<td>Secondary Email</td>
<td><inputtype="text"name="email"/></td></tr>
<tr>
<td>Student Name</td>
<td><inputtype="submit"value="submit"/></td></tr
>
</table>
</form>
```

2.show.jsp

```
<body>
<h1>Java Training Center</h1>
<table>
<tr><td>Student Name</td>
<td>${param.sname }</td></tr>

<tr><td>Primary Email</td>
<td>${param.email }</td></tr>

<tr><td>Primary Email</td>
<td>${param.email }</td></tr>

<tr><td>Primary Email</td>
<td>${paramValues.email[0] }</td></tr>

<tr><td>Secondary Email</td>
<td>${paramValues.email[1] }</td></tr>

<tr><td>Host</td>
<td>${header.host }</td></tr>

<tr><td>Referer</td>
<td>${header.refere }</td></tr>

<tr><td>JSessionID Cookie</td>
<td>${cookie.email.value }</td></tr>
```

```
</body>
</html>

3.web.xml
<welcome-file-list>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<context-param>
<param-name>state</param-name>
<param-value>Noida</param-value>
</context-param>
<context-param>
<param-name>city</param-name>
<param-value>Noida</param-value>
</context-param>
<servlet>
<servlet-name>test</servlet-name>
<servlet-
class>com.jtcindia.TestServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>test</servlet-name>
<url-pattern>/test.jtc</url-pattern>

</servlet-mapping>
```

```
<tr><td>Phone Cookie</td>
<td>${cookie.phone.value }</td></tr>

<tr><td>CP:city</td>
<td>${initParam.state }</td></tr>

<tr><td>CP:city</td>
<td>${initParam.city }</td></tr>

<tr><td>using PageContext</td>
<td>${pageContext.session.id }<br/>
${pageContext.request.remoteAddr }<br/>
${pageCotext.request.method }<br/>
${pageContext.request.requestURI }<br/>
${pageContext.response.contentType
}<br/>
</td></tr>
</table>
</body>
</html>

4.TestServlet.java
package com.jtcindia;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

publicclassTestServletextends
HttpServlet {
    @Override
    protectedvoid
service(HttpServletRequest req,
HttpServletResponse res)
                    throws
ServletException, IOException {
            //1. Collect parameters
            String
sn=req.getParameter("sname");
            String
em=req.getParameter("email");
            String
ems[]=req.getParameterValues("email");
            System.out.println(sn);
```

```
                System.out.println(em);
                for(String str:ems){

        System.out.println(str);
                }
                //2.Collect Header
                String
ho=req.getParameter("host");
                String
re=req.getParameter("refere");
                String
al=req.getParameter("accept-language");
                String
ae=req.getParameter("accesp-encoding");
                System.out.println(ho);
                System.out.println(re);
                System.out.println(al);
                System.out.println(ae);
                //3.Collect Cookies
                Cookie
ck[]=req.getCookies();
                for(Cookie c:ck){

        System.out.println(c.getName()+"-
--"+c.getValue());
                }
                Cookie ck1=new
Cookie("email","som@jtc.com");
                Cookie ck2=new
Cookie("email","123456");
                res.addCookie(ck1);
                res.addCookie(ck2);
                RequestDispatcher
rd=req.getRequestDispatcher("show.jsp")
;
                rd.forward(req, res);
        }
}
```

pageScope, requestScope, sessionScope, applicationScope

- In servlet class and Tag handler class , you can store the data as attribute and in jsp , you can collect the attribute data.
- In servlet class , you can store the data as attribute.
- In jsp, you can collect the attribute data and display that data.
- You can store the followinng type of data as attribute in the required scope.

You can store the following type of data as attribute in the required scope.

1. Strings,Wrappers and Date.
2. Collection of Strings,Wrappers and Date.
3. Collection of Collections (*JSTL)

4. Custom or User defined class object(Customer, Book, Student)
5. Collection of User defined class object
6. Map object
7. Collection of Map object (*JSTL)

1)Strings, Wrappers and Date.

In Servlet

```
req.setAttribute("EM","som@jtc");
ses.setAttribute("PH",new Long(12345));
ctx.setAttribute("DOB",new Date());
```

In JSP

```
${EM}
${PH}
${DOB}
```

If you are not specifying the scope then it will check in all the scope if found it will display the corresponding data. First it will check in page, request, session and then in context.

OR

```
${requestScope.EM}
${sessionScope.PH}
${applicationScope.DOB}
```

2) Custom or User defined class object(Customer)

In Servlet

```
class Customer{
cid
cname
email
Address address;
}
class Address{
street
city
}
ses.setAttribute("CUST",cust);
```

In JSP

```
Cid : ${CUST.cid}
Cname : ${sessionScope.CUST.cname}
```

```
Email : ${sessionScope.CUST.email}
Street: ${sessionScope.CUST.address.street}
City: ${sessionScope.CUST.address.city}
```

3) Collection of Strings, Wrappers and Date.

In Servlet

```
ArrayList al=new ArrayList();
al.add(new Integer(123));
...
String str[]={"dd","ss",...};
req.setAttribute("AL",al);
ses.setAttribute("STR",str);
```

In JSP

```
${requestScope.AL[0]}
${requestScope.AL[1]}
${sessionScope.STR[0]}
${sessionScope.STR[1]}
..
```

5) Collection of User defined class object

In Servlet

```
ArrayList al=new ArrayList();
...
al.add(cust1);
al.add(cust2);
ses.setAttribute("AL",al);
```

In JSP

```
${sessionScope.AL[0]}
Cid : ${AL[0].cid}
Cname : ${sessionScope.AL[0].cname}
Email : ${sessionScope.AL[0].email}
Street: ${sessionScope.AL[0].address.street}
City: ${sessionScope.AL[0].address.city}
```

6) Map object

In Servlet

```
HashMap hm=new HashMap();
hm.put("sid","11");
hm.put("sname","som");
..
ses.setAttribute("HM",hm);
```

In JSP

Sid : ${sessionScope.HM["sid"]}
Sname : ${sessionScope.HM["sname"]}

Jtc9:

1.index.jsp

```
<%@pagelanguage="java"contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Java Training Center</h1>
<formaction="test">
<table>
<tr>
<td>Name:<inputtype="text"name="name"></td>
</tr>
<tr>
<td><inputtype="submit"value="submit"/>
</td>
</tr>
</table>
</form>

</body>
</html>
```

2.show.jsp

```
<%@pagelanguage="java"contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
```

3.web.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<web-appxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns="http://java.sun.com/xml/ns/javaee"xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"id="WebApp_ID"version="3.0">
<display-name>Jtc5</display-name>
<welcome-file-list>
<welcome-file>index.jsp</welcome-file>
</welcome-file-list>
<servlet>
<servlet-name>test</servlet-name>
<servlet-class>com.scope.ScopeServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>test</servlet-name>
<url-pattern>/test</url-pattern>

</servlet-mapping>
</web-app>
```

4.Account.java

```
package com.scope1;

      publicclass Account {

            privateintaccno;

            private String atype;
            privateintbal;
            private Address address;

public Account(intaccno,String atype,intbal,Address address){



                  this.accno=accno;
                  this.atype=atype;
```

```
</head>
<body>

<h1>JAva Training center </h1>
<table><tr>
<td>1.String,Wrapper and Date.</td>
<td>${requestScope.EM }<br/>
${sessionScope.PH }<br/>
${applicationScope.DOB }<br/>
</td></tr>

<tr>
<td>2.Collection of String,Wrapper and
Date</td>
<td>${requestScope.AL[0] }<br/>
${requestScope.AL[1] }<br/>
${requestScope.AL[2] }<br/>
${requestScope.AL[3] }<br/>
${requestScope.AL[4] }<br/>
${sessionScope.STR[0] }<br/>
${sessionScope.STR[1] }<br/>
${sessionScope.STR[2] }<br/>
${sessionScope.STR[3] }<br/>
</td></tr>
<tr>
<td>3.Collection of String,Wrapper and
Date</td>
<td>coming soon<br/></td>
</tr>
<tr>
<td>4.Custom or User define class
object</td>
<td>
Cid:${CUST.cid}<br/>
Cname:${CUST.cname }<br/>
Email:${CUST.email }<br/>
Phone:${CUST.phone }<br/>
Accono:${CUST.acc.accono }<br/>
Atype:${sessionScope.CUST.acc.atype
}<br/>
Bal:${sessionScope.CUST.acc.bal }<br/>
Street:${sessionScope.CUST.acc.street
}<br/>
City:${sessionScope.CUST.acc.city }<br/>
</td>
</tr>

<tr>
<td>5.Collection of User define class
Object</td>
<td>
Cid:${CS[0].cid}<br/>
```

```
                this.bal=bal;
                this.address=address;

        }

publicint getAccno() {
returnaccno;
        }

publicvoid setAccno(intaccno) {
      this.accno = accno;
        }

public String getAtype() {
returnatype;
        }
publicvoid setAtype(String atype) {
this.atype = atype;
        }

publicint getBal() {
      returnbal;
        }

publicvoid setBal(intbal) {
      this.bal = bal;
        }
public Address getAddress() {
      returnaddress;
        }

publicvoid setAddress(Address address) {
      this.address = address;
        }

        }

5.Customer.java
package com.scope1;

//import com.scope.Account;

      publicclass Customer {
            privateintcid;
            private String cname;
            private String email;
            privatelongphone;
            private Account acc;


            public Customer(intcid,String
cname,String email,longphone,Account acc){
```

```
Cname:${sessionScope.CS[0].cname }<br/>
Email:${sessionScope.CS[1].email }<br/>
Phone:${sessionScope.CS[2].phone }<br/>
Accono:${sessionScope.CS[3].account.accon
o }<br/>
Atype:${sessionScope.CS[4].account.atype
}<br/>
Bal:${sessionScope.CS[5].account.bal
}<br/>
Street:${sessionScope.CS[6].account.stree
t }<br/>
City:${sessionScope.CS[7].account.city
}<br/></td>
</tr>

<tr>
<td>6.Map Object</td>
<td>
Sid:${sessionScope.HM["sid"] }<br/>
Sname:${sessionScope.HM["sname"] } <br/>
Email:${sessionScope.HM["email"] }<br/>
</td>
</tr>

<tr>
<td>7.Collection of Map Object(*jstl)
<td>coming soon</td>

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

7.TestServlet.java

```
package com.scope;
import com.scope1.*;


import java.io.IOException;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;

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 javax.servlet.http.HttpSession;
```

```
                super();

                this.cid=cid;
                this.cname=cname;
                this.email=email;
                this.phone=phone;
                this.acc=acc;


        }


        publicint getCid() {
                returncid;
        }


        publicvoid setCid(intcid) {
                this.cid = cid;
        }


        public String getCname() {
                returncname;
        }


        publicvoid setCname(String cname)
{
                this.cname = cname;
        }


        public String getEmail() {
                returnemail;
        }


        publicvoid setEmail(String email)
{
                this.email = email;
        }


        publiclong getPhone() {
                returnphone;
        }


        publicvoid setPhone(longphone) {
                this.phone = phone;
        }
```

```
public class ScopeServlet extends HttpServlet {
	public void service(HttpServletRequest req,HttpServletResponse res) throws ServletException, IOException{
		HttpSession hs=req.getSession();
		ServletContext sc=hs.getServletContext();

		//1.String wrapper

	req.setAttribute("EM","raj@abc.org");
		hs.setAttribute("PH",new Long(123344));
		sc.setAttribute("DOB",new Date());

		//2.Collection of arraylist

		ArrayList<Integer> al=new ArrayList<Integer>();
		al.add(new Integer(11));
		al.add(new Integer(22));
		al.add(new Integer(33));
		al.add(new Integer(44));
		al.add(new Integer(55));
		req.setAttribute("AL",al);
		String str[]={"dd","ss","aff","raj"};
		sc.setAttribute("STR",str);

		//3.Collection of collection (jstl)

		ArrayList<ArrayList> al1=new ArrayList<ArrayList>();
		al1.add(al);
		al1.add(al);
		al1.add(al);

		req.setAttribute("AL1",al1);

		//4.Collection of User Define Class Object

		Address add=new Address("sec-2","Noida");
		Account acc=new Account(11,"rajjj",335,add);
		Customer cust=new Customer(22, "abc","null@kwdk",873999, acc);
```

```
	public Account getAccount() {
		returnacc;
	}


	publicvoid setAccount(Account account) {
		this.acc = acc;
	}
	}
6.Address.java
package com.scope1;


	publicclass Address {

		private String street;
		private String city;

		public Address(String street,String city){



		this.street=street;
		this.city=city;


		}
		public String getStreet() {
			returnstreet;
		}

		publicvoid setStreet(String street) {
			this.street = street;
		}

		public String getCity() {
			returncity;
		}

		publicvoid setCity(String city) {
			this.city = city;
		}


}
```

```
                sc.setAttribute("CUST",cust);

                //5.Collection of user define class
object

                ArrayList<Customer> ac=new
ArrayList<Customer>();
                ac.add(cust);
                ac.add(cust);
                ac.add(cust);
                ac.add(cust);
                ac.add(cust);
                sc.setAttribute("CS",ac);

                //6 Map Object

                HashMap<String, String> hm=new
HashMap<String,String>();
                hm.put("sid","11");
                hm.put("sname","raj");
                hm.put("email","raj@.org");
                sc.setAttribute("HM",hm);

                //7 Collection of Map
Object(JSTL)


          ArrayList<HashMap<String,String>>
al2=new ArrayList<HashMap<String,String>>();
                al2.add(hm);
                al2.add(hm);
                al2.add(hm);
                req.setAttribute("AL2",al2);

                RequestDispatcher
rd=req.getRequestDispatcher("/show.jsp");
                rd.forward(req, res);

          }

}
```

JSTL

JSTL (JSP Standard Tag Library)
JSTL is collection of utility tags which can be used by developer in jsp.

To use JSTL tags, you have to do the following.

1) Copy jstl.jar and standard.jar to your project WEB-INF/lib folder.
2) Use the following taglib directive in any jsp.
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

List of JSTL Core Tags

1. <c:out>
2. <c:set>
3. <c:remove>
4. <c:if>
5. <c:choose>
6. <c:when>
7. <c:otherwise>
8. <c:forEach>
9. <c:import>
10. <c:redirect>
11. <c:param>
12. <c:catch>
13. <c:url>

1) <c:out>
out tag is used to display data to JSP output stream.
attributes : value.

Usage:
<c:out value="msg"/>
<c:out value="${msg}"/><==> ${msg}

2) <c:set>
set tag is used to set the attribute in the required scope.
attributes : var, value, scope

Usage:
<c:set var="em" value="som@jtc" scope="session"/>

3) <c:remove>
remove tag is used to remove the attribute from the required scope.
attributes : var, scope

Usage:
<c:remove var="em" scope="session"/>

4) <c:if>
if tag is used to perform simple conditional checks.
attributes : test

Usage:
<c:if test="${sname eq 'som'}">

Your name is som
</c:if>
<c:if test="${sname ne 'som'}">
Your name is Not som
</c:if>

5) <c:choose>, 6) <c:when> and 7) <c:otherwise>

These tags are used to perform simple conditional checks like switch statement

attributes :
choose - no attributes available
when - test
otherwise - no attributes available

Usage:

```
<c:choose>
<c:when test="${sid eq 99}">
id is 99
</c:when>
<c:when test="${sid eq 88}">
id is 88
</c:when>
<c:otherwise>
id is unknown
</c:otherwise>
</c:choose>
```

8) <c:forEach> :

forEach tag is used to access the elements of collection or array.

attributes: var , items , start , end , step , varStatus

Usage:
Using forEach tag ,You can access the following type of data.

1. Collection of Strings, Wrappers and Date.
2. Collection of Collections
3. Collection of User defined class object
4. Map objects
5. Collection of Map objects

1) Collection of Strings,Wrappers and Date.

In Servlet

```
ArrayList al=new ArrayList();
al.add(new Integer(123));
```

```
...
String str[]={"dd","ss",...};
req.setAttribute("AL",al);
ses.setAttribute("STR",str);
```

In jsp

```
<c:forEach var="x" items="${requestScope.AL}">
${x} <br/>
</c:forEach>

<c:forEach var="x" items="${sessionScope.STR}">
${x} <br/>
</c:forEach>
```

2) Collection of Collections

In Servlet

```
ArrayList al=new ArrayList();
ArrayList al1=new ArrayList();
al1.add(new Integer(123));
ArrayList al2=new ArrayList();
al2.add(new Integer(123));
al.add(al1);
al.add(al2);
ses.setAttribute("AL",al);
```

In jsp

```
<c:forEach var="list" items="${sessionScope.AL}">
<c:forEach var="x" items="${list}">
${x} <br/>
</c:forEach>
</c:forEach>
```

3) Collection of User defined class object

In Servlet

```
ArrayList al=new ArrayList();
...
al.add(cust1);
al.add(cust2);
ses.setAttribute("AL",al);
```

In JSP

```
<c:forEach var="cust" items="${sessionScope.AL}">
Cid : ${cust.cid}
Cname : ${cust.cname}
Email : ${cust.email}
Street: ${cust.address.street}
```

City: ${cust.address.city}
</c:forEach>

4) Map object

In Servlet
HashMap hm=new HashMap();
hm.put("sid","11");
hm.put("sname","som");..
ses.setAttribute("HM",hm);

In JSP
<c:forEach var="key" items="${sessionScope.HM}">
${key} : ${sessionScope.HM['${key}']}
</c:forEach>

5) Collection of Map objects

In Servlet
HashMap hm=new HashMap();
hm.put("sid","11");
hm.put("sname","som");
..
ArrayList al=new ArrayList();
al.add(hm);
al.add(hm);
ses.setAttribute("AL",al);

In JSP
Sid : ${sessionScope.HM["sid"]}
Sname : ${sessionScope.HM["sname"]}
<c:forEach var="map" items="${sessionScope.AL}">
<c:forEach var="key" items="${map}">
${key} : ${map['${key}']}
</c:forEach>
</c:forEach>

Pagination :

In servlet
ArrayList al=new ArrayList();
al.add(new Integer(123));
...
req.setAttribute("AL",al);
int start=1;
int end=100;
ses.setAttribute("START",start);
ses.setAttribute("END",end);

```
Inetegr i1=(Integer)ses.getAttribute("START");
Inetegr i2=(Integer)ses.getAttribute("END");
int start=i1.intValue()+100;
int end=i2.intValue()+100;
ses.setAttribute("START",start);
ses.setAttribute("END",end);
<c:forEach var="x" items="${requestScope.AL}" start="${START}" end="${END}">
${x} <br/>
</c:forEach>
```

********JSTLJtc3 (WB) :Explanation********

********JSTLJtc4 (WB) :Explanation********

9) <c:import>

import tag is used to include the resource(html or jsp).
attributes: url

Usage:

```
<c:import url="header.jsp"/>
<c:import url="http://www.google.co.in"/>
<c:import url="http://www.jtcindia.org/contactus.jsp"/>
```

<jsp:include> can include the resources which are in same server
where as <c:import> can include the resources which are in same server and different server.

10) <c:redirect>

redirect tag is used to redirect the request to the specified resource(html or jsp).
attributes: url

Usage:

```
<c:redirect url="header.jsp"/>
<c:redirect url="http://www.google.co.in"/>
<c:redirect url="http://www.jtcindia.org/contactus.jsp"/>
```

<jsp:forward> can forward request to resources which are in same application where as <c:redirect> can forward request to the resources which are in same application and different application running in the same server or different server.

11) <c:param>

param tag is used to define parameters.
attributes : name , value

Usage :

<c:param name="email" value="som@jtcindia.org"/>
<c:param> must be used along with <c:import> and <c:redirect>

12)<c:catch>

catch tag is used to implement try catch functionality.

Usage:
<c:catch>
....
</c:catch>

13)<c:url>

url tag is used to encode the url with sessionid.
attributes: value

Usage :

<c:url value="hello.jsp"/>
 Click Here
<a href='<c:url value="hello.jsp"/>'> Click Here
<a href='<%= response.encodeURL("hello.jsp")%>'> Click Here

Note :
${entry} and ${sessionScope.HM['${entry}']} are providing Map.Entry class objects.

<c:forEach var="entry" items="${sessionScope.HM}">
${entry.key} : ${entry.value}
</c:forEach>

Jtc10:

1. index.jsp (same as Jtc9)	2. show.jsp
3. web.xml(same as Jtc9)	4. TestServlet.java same as Jtc9
5. Address.java(same as Jtc9)	6. Customer.java same as Jtc9
7. Account.java(same as Jtc9)	

1. Show.jsp <%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%> <%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %> <!DOCTYPE html PUBLIC "-//W3C//DTD	</td> </tr> <tr> <td>3.Collection of Collections:</td> <td><table bgcolor="#c3d599" style="font-size: 26; color: red;">

```
HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<center>
<h1>Java training Center</h1>
<table border="2" bgcolor="red">
<tr>
<td>1. String wrapper and Date</td>
<td><c:out value="${requestScope.EM }"/><br>
                        ${sessionScope.PH }<br>
                        ${applicationScope.DOB
}<br>
                        </td>
                        </tr>

                        <tr><td>2.Collection of
String Wrraper and Date:</td>
                        <td><table
bgcolor="#c3d559" style="font-size: 26;">
                        <tr>
                        <c:forEach var="x"
items="${requestScope.AL }"><td>${x }</td>
                        </c:forEach></tr>
                        </table>
                        <table bgcolor="wheet"
style="font-size: 26; color: red;">
                        <tr><c:forEach var="i"
items="${sessionScope.STR }"><td>${i
}</td></c:forEach>
                        </tr>
                        </table>
                        <c:forEach var="list"
items="${requestScope.AL1 }">
                        <tr>
                        <c:forEach var="x" items="${list
}"><td>${x }</td>
                        </c:forEach></tr></c:forEach>

                        </table></td>
                        </tr>

                        <tr>
                        <td>4.Custmer or user defined
class object</td>
                        <td>Cid:${sessionScope.CUST.id
}<br>Email:${sessionScope.CUST.email }<br>

        Phone:${sessionScope.CUST.phone }<br>

        Accno:${sessionScope.CUST.account.accno }<br>

        Atype:${sessionScope.CUST.account.atype }<br>

        Bal:${sessionScope.CUST.account.bal }<br>

        Street:${sessionScope.CUST.account.address.street
}<br>

        City:${sessionScope.CUST.account.address.city
}<br></td>
                        </tr>

                        <tr>
                        <td>5.collection or user defined
class object</td>
                        <td><table bgcolor="#c3d599"
style="font-size: 26; color: red;">
                        <tr>
                        <td>Cid</td>
                        <td>Cname</td>
                        <td>Email</td>
                        <td>Phone</td>
                        <td>Accno</td>
                        <td>Atype</td>
                        <td>Bal</td>
                        <td>Street</td>
                        <td>City</td>
                        </tr>
                        <c:forEach  var="cust"
```

```
items="${sessionScope.CS }">
                        <tr>
                        <td>${cust.Cid }</td>
                        <td>${cust.cname }</td>
                        <td>${cust.email }</td>
                        <td>${cust.account.accno }</td>
                        <td>${cust.account.atype }</td>
                        <td>${cust.account.bal }</td>
                        <td>${cust.account.address.street
}</td>
                        <td>${cust.account.address.city
}</td>
                        </tr>
                        </c:forEach>
                        </table></td></tr>



                        <tr><td>6.Map object</td>
                        <td><table bgcolor="red"
style="font-size: 26; color: red;">
                        <tr>
                        <td>Map key</td>
                        <td>Map value</td>
                        </tr><c:forEach var="entry"
items="${requestScope.HM }">
                        <tr>
                        <td>${entry.key }</td>
                        <td>${entry.value
}</td></tr></c:forEach></table></td></tr>



                        <tr>
                        <td>7.collection of map
Object</td>
                        <td><table bgcolor="wheet"
style="font-size: 26; color: red;">
                        <c:forEach var="map"
items="${requestScope.AL2 }">
                        <tr><td><table
bgcolor="#c3d599" style="font-size: 26; color: red;"><tr>
                        <td>Map key</td>
                        <td>Map value</td>
                        </tr>
                        <c:forEach  var="entry"
items="${map }"><tr>
                        <td>${entry.key }</td>
                        <td>${entry.value }</td>
                        </tr></c:forEach></table>
```

	</td></tr> </c:forEach> </table></td></tr> </table> </body> </html>

Jtc11:

1. Header.jsp	2. Footer.jsp	3. Test.jsp
4. Home.jsp	5. Java.jsp	6. Jdbc.jsp
7. Other.jsp		

1.Header.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>
${param.companyName }
</h1>
<h1>
<c:outvalue="${param.companyName }"/>
</h1>
</body>
</html>
```

4.home.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Hello guys No Gaurantee</h1>
<%
String str="JAVA,JDBC,Servlet,JSP";
request.setAttribute("STR",str);
%>
<c:forTokensitems="${STR
}"delims=","var="c"><br/>${c }</c:forTokens>
<c:forEachbegin="1"end="25"var="i"><br/>${i
}</c:forEach>
```

2.footer.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>All Rights
Reserved.${param.companyName }.2011</h1>
</body>
</html>
```

3.test.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
</head>
<body>
<c:importurl="header.jsp">
<c:paramname="companyName"value="Java
Training Center"/>
</c:import>
<formaction="home.jsp"method="post">
<h2> Enter Course Name</h2>
<inputtype="text"name="courseName"/>
<h2>Enter Student Name</h2>
<inputtype="text"name="studentName"/>
<inputtype="submit"value="submit">
```

```
<c:setvar="CN"value="${param.courseName
}"scope="session"/>

<c:choose>
<c:whentest="${CN eq 'Java' }">
<c:redirecturl="java.jsp">
<c:paramname="stuName"value="${param.studentName
}"/>
<c:paramname="couName"value="${param.courseName
}"/>
</c:redirect>
</c:when>
<c:whentest="${CN eq 'Jdbc' }">
<c:redirecturl="jdbc.jsp">
<c:paramname="stuName"value="${param.studentName
}"/>
<c:paramname="couName"value="${param.courseName
}"/>
</c:redirect>
</c:when>

<c:otherwise>
<c:redirecturl="other.jsp">
<c:paramname="stuName"value="${param.studentName
}"/>
<c:paramname="couName"value="${param.courseName
}"/>
</c:redirect>
</c:otherwise>
</c:choose>
</body>
</html>
```

```
</form>
<c:importurl="footer.jsp">
<c:paramname="companyName"value="Java
Training Center"/>
</c:import>

</body>
</html>
```

5.java.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
</head>
<body>
<c:importurl="header.jsp">
<c:paramname="companyName"value="Java
Training Center"/>
</c:import>
<h1>This is java.jsp<br>
Hello ! ${param.stuName },<br>
You Have selected ${param.couName }</h1>
<c:importurl="footer.jsp">
<c:paramname="companyName"value="Java
Training Center"></c:param>
</c:import>
</body>
</html>
```

6.jdbc.jsp

```
<html>
<head>
<metahttp-equiv="Content-
Type"content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
</head>
<body>
<c:importurl="header.jsp">
<c:paramname="companyName"value="Java
Training Center"/>
</c:import>
<h1>This is jdbc.jsp<br>
Hello ! ${param.stuName },<br>
You Have selected ${param.couName }</h1>
<c:importurl="footer.jsp">
<c:paramname="companyName"value="Java
Training Center"></c:param>
</c:import>
</body>
</html>
```

	7.other.jsp

Custom tags

- When you want to display the dynamic content in the JSP then you can use EL Expression and JSTL Tags.
- If these two are not enough for your application requirements then you can devlop your own tags called as custom Tags.
- When you devlop a custom tag then can be reused across multiple JSPs.

Steps to develop the custom tags:

- Identify the information related to the tag which you want to devlop includes
  - Name of the tag
  - Body-content
  - Attributes allowed for the tag
- Write the tag handller class with the following steps:
  - Define java class by extending TagSupport or BodyTagSupport class.
  - Define the variable for the attributes defined for the tag.
  - Define the setter method for the variables.
  - Override the required lifecycle methods.
- Place one tld file information in the tld file which includs
  - Uri
  - Name of the tag
  - Body-content
  - Tag-class
  - Attributes allowed for the tag etc.

Step to use Custom tag:

- Use the taglib directive to use the custom tags in jsp.
- Taglib directive has two attributed called prefix and uri.
- URI of the taglib directive must match with the URI specified in the TLD file.
- Prefix of the taglib directive can be anything as you like.
- Refere the tag in the JSP using prefix.

Ex:

```
<jtc:showMessage sname='${param.studname}' email='${param.studname}'/>
        <jtc:showMessage sname='som' email='som@jtc.com'/>
```

Note:

- If the value of <rtexpvalue>is true the you can provide the value for the attribute dynamically using scriptlet or EL expression.

```
<jtc:showMessage sname="<%=request.getParameter("name">%>" email="${param.email}"/>
```

Custom Tag Processing Flow

- Whenever tag is encountered in the JSP, then container takes tag name and prefix.
- Container checks whether any taglib directive is available whose prefix is matching with prefix of the tag encountered.
- If no matching taglib directive is available then that tag will be ignored.
- If matching taglib directive is available then URI of matching taglib directive will be taken.
- Container checks whether any tld file is available whose URI is matching with URI of the taglib directive indentified.
- If matching tld file is found then container checks tag descrption in that tld file by checking the following.
    - Checks whether tag name is found in that tld or not.
    - If tag name is not found then error message will be given
        - (no tag “showMessage” defined in tag library imported with prefix “JTC”)
    - If tag name is found then checks whether the attributes are used correctly or not.
    - If attribute names are not used correctly thenn error message will be given
        - Underfined attribute name “phone”
        - Missing required attribute “sname”
    - If attributes are used correctly then tag Tag Handdler class will be taken.
- Container starts the tag Handler class lifecycle.
- Tag Handler is the class That is responsible to process the custom tags.
- Once the tag Handler class lifcycle is completed then Tag processing is completed.

<u>Lifecycle methods of tag Handdler class which is extending TagSupport class:</u>

- Public int doStartTag()
- Public int doAfterBody()
- Public int doEndTag()
- void setParent(Tag)
- Tag getParent()
- void setPageContext(PageContext)
- PageContext getPageContext()

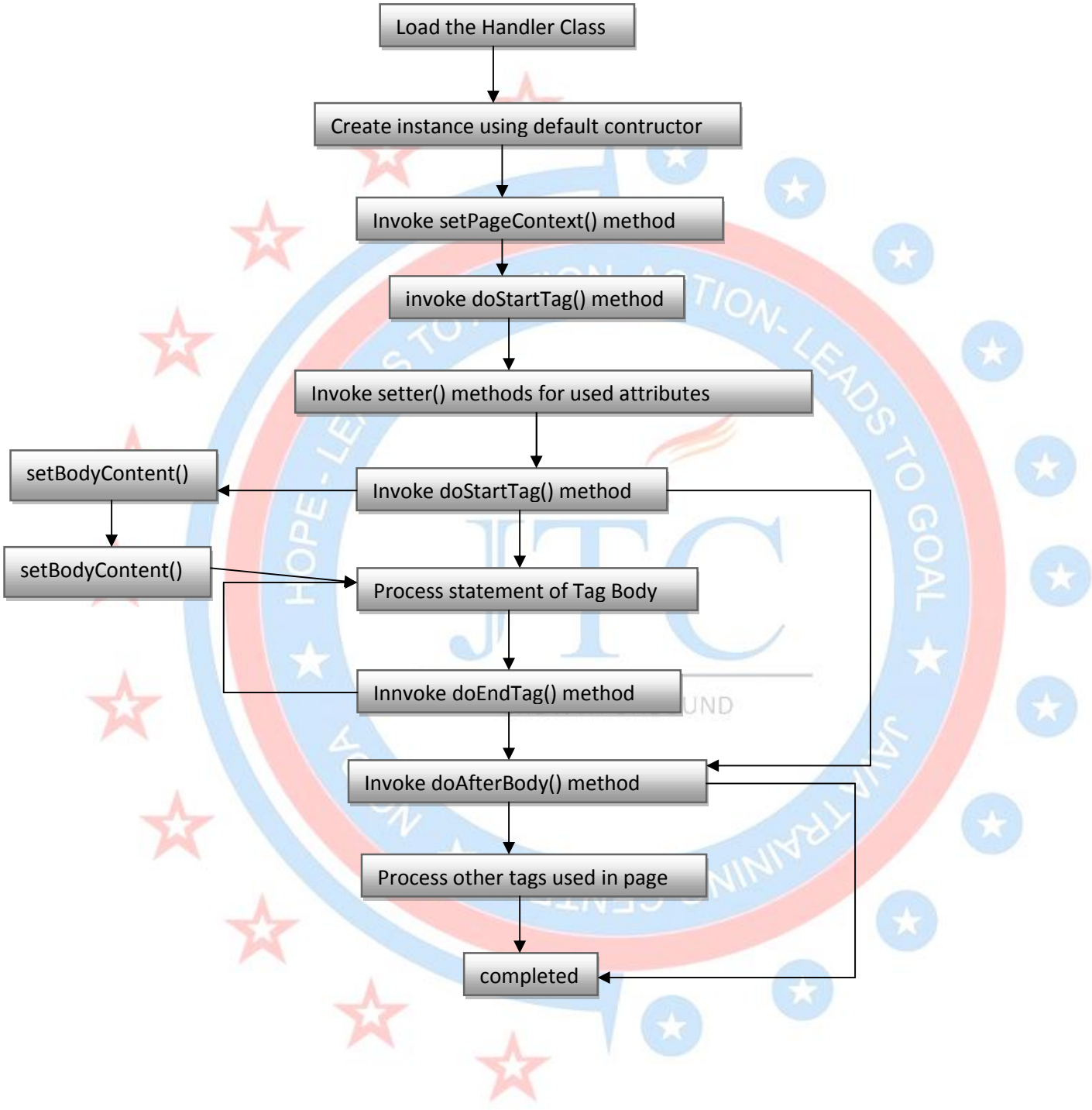
Using BodyTagSupport

- You can devlop the tags with and without body using TagSupport as a super class.
- Using TagSupport, you can devlop all types of tags like simple tags, nested tags, tags with body,looping tags etc.
- BodyTagSupport should be used only in one special case i.e if you want get the body into your control and you want to modify the body as per the requirement, then only use BodyTagSupport as super class to your Tag Handler class.
- If tag handdler class is extending BodyTagSupport then you can use following two extra lifecycle method in the TagHandler class.
  - Public void setBodyContent(BodyContent etc)
  - Public void doInitBody()
- You need to return following value from doStartTag() to excute the setBodyContent() method
  - EVAL_BODY_BUFFERED
- This is the default result from doStartTag() in the case of BodyTagSupport class.
- You can use the following method to get the body content in the tag handler class
  Public BodyContent getBodyContent()

Lifecycle methods of Tag Handler class which is extending TagSupport class:

- Public int doStartTag()
- Public int doAfterBody()
- Public int doEndTag()
- void setParent(Tag)
- Tag getParent()
- void setPageContext(PageContext)
- PageContext getPageContext()

Life Cycle of the Tag handdler that extends BodyTagSupport



doStartTag() BodyTagSupport		
Posible return values	SKIP_BODY EVAL_BODY_INCLUDE EVAL_BODY_BUFFERED	SKIP_BODY EVAL_BODY_INCLUDE
Default return value from the implementing	EVAL_BODY_BUFFERED	SKIP_BODY
Number of times its can be called (per tag invocation from a jsp)	Exactly Once	Exactly Once
doAfterBody()		
BodyTagSupport		
Possible return values	SKIP_BODY	SKIP_BODY EVAL_BODY_AGAIN
Default return value from the implementing class	SKIP_BODY	SKIP_BODY
Number of times its can be called (per tag invocation from a JSP)	Zero to Many	Zero to money
	doEndTag()	
Possible return values	SKIP_PAGE EVAL_PAGE	SKIP_PAGE EVAL_PAGE
Default return value from the implementing class	EVAL_Once	EVAL_PAGE
Number of times its can be called (per tag invocation from a JSP)	Exactly Once	Exactly Once
	doInitBody() and setBodyContent()	
Circumstance under which they can be called and number of times per tag invocation	Exactly once and only if doStartTag() returns EVAL_BODY_BUFFERED	NEVER