UNIT-4

Introduction to JSP: The Anatomy of a JSP Page, JSP Processing, Declarations, Directives, Expressions, Code Snippets, implicit objects, Using Beans in JSP Pages, Using Cookies and session for session tracking, connecting to database in JSP.

Introduction to JSP

- JSP is one of web technology from Sun Microsystems.
- A JSP is server side piece of code that enhances the functionality of web server.
- It provides dynamic web content i.e. dynamic web page.
- In servlets, HTML code is written inside Java methods, but in JSP, java code is written in HTML tags.

Introduction to JSP

- Limitations of Servlets:

- Servlets alone can do several task (Acceptance of request, processing request, handling business logic, generation of response), but poor in presentation.
- For servlet based applications, knowledge of Java as well as HTML is necessary.
- While developing web application, if look & feel of web needs to be changed then the entire code is to be changed and must be recompiled.
- They do not support web page development tools. To use them we need to change embedded HTML manually, which is time consuming, error prone and complicated process.

Introduction to JSP

JSP is one technology in which request processing, business logic and presentations are separated out.

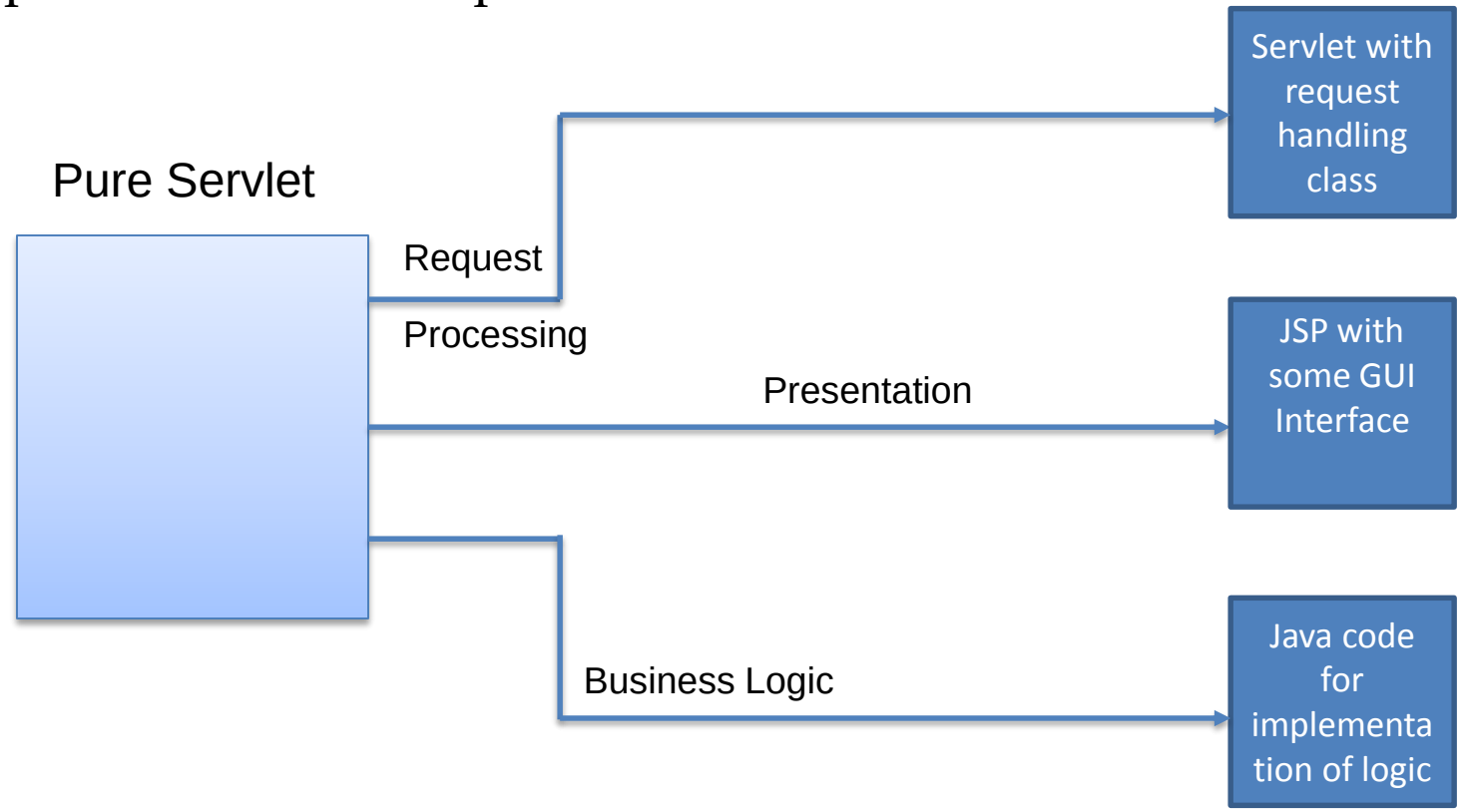
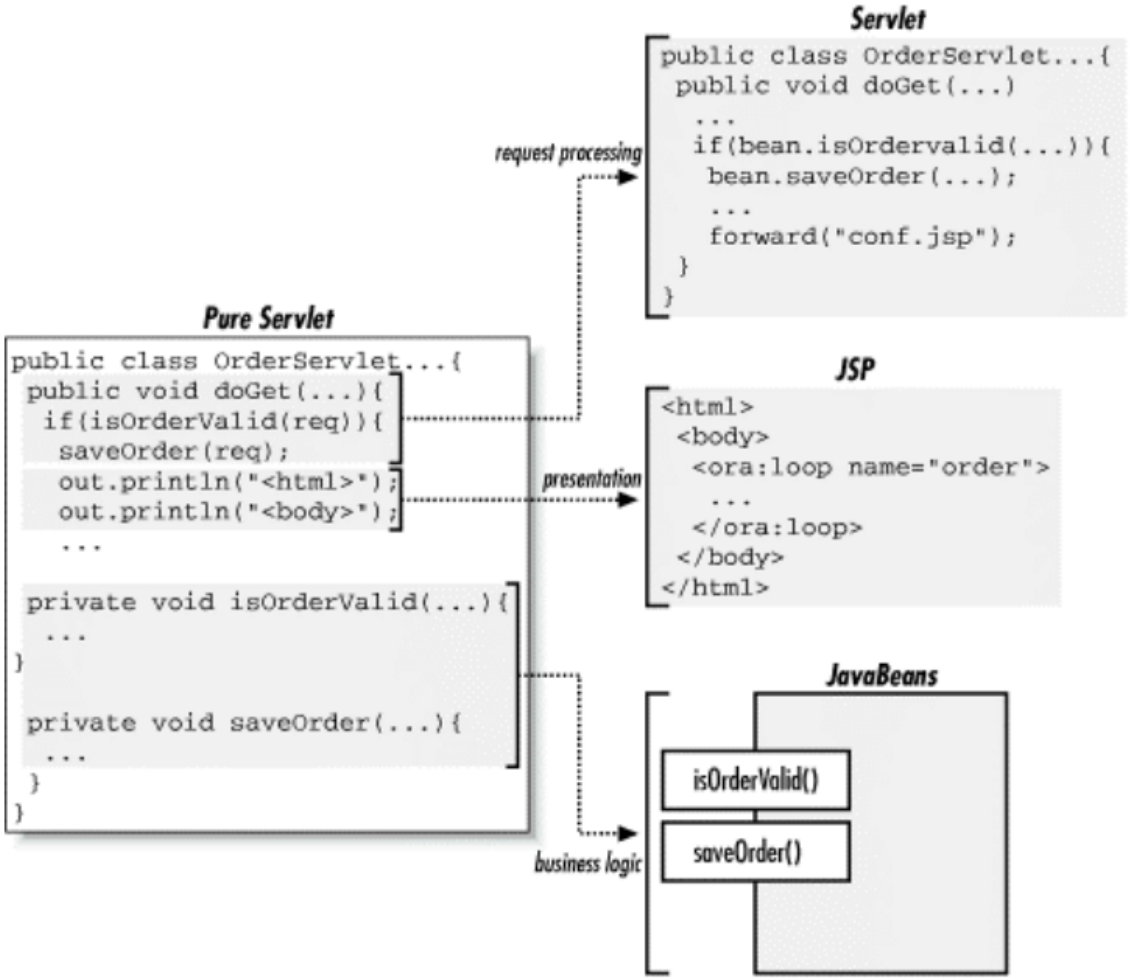


Fig: Separation request processing, presentation & business logic

Introduction to JSP



The Anatomy of JSP Page

A JSP page is simply a regular web page with JSP elements for generating the parts of the page that differ for each request.

Everything in the page that is not a JSP element is called template text .

Template text can really be any text: HTML, WML, XML, or even plain text.

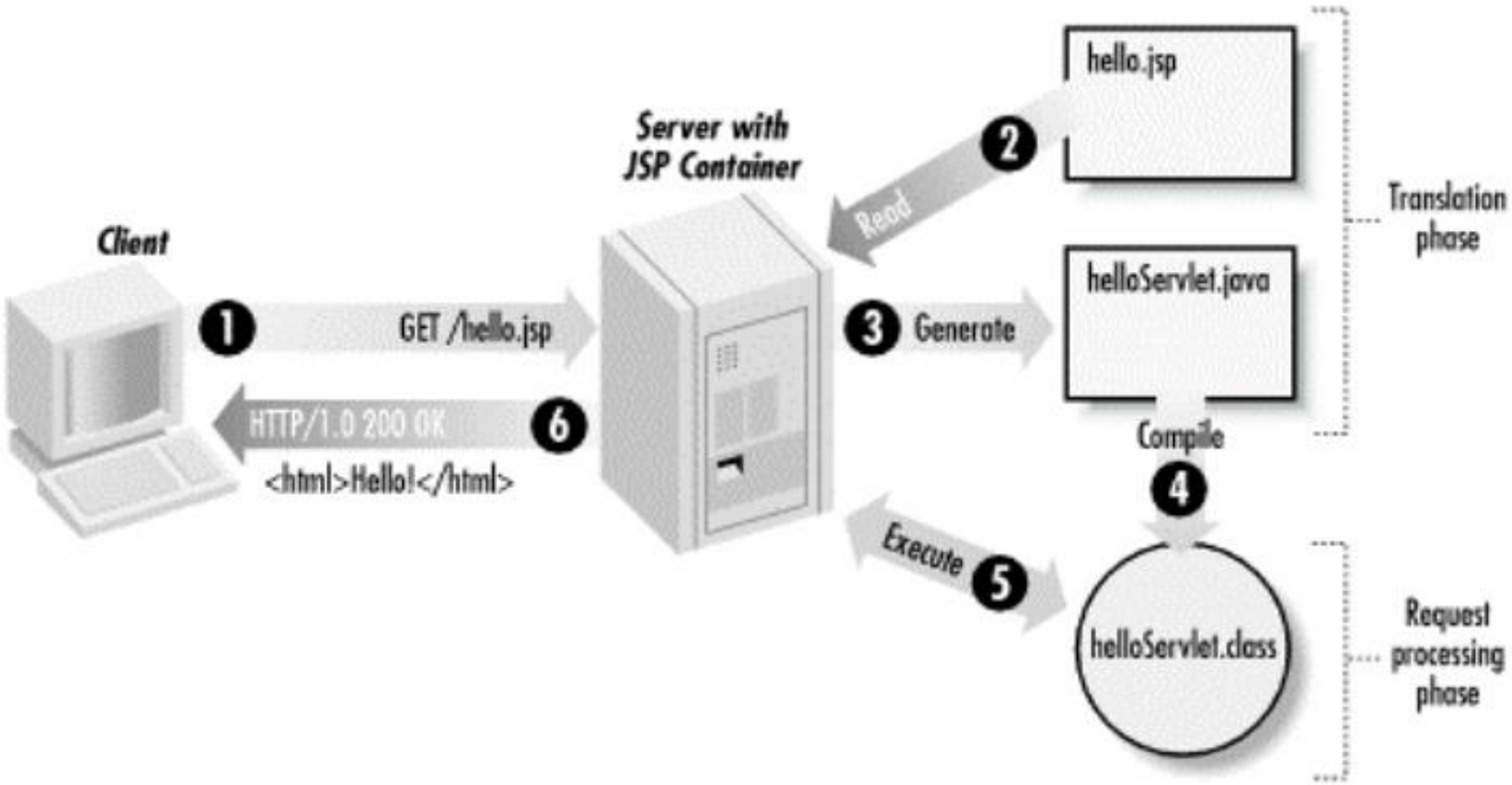
```
<%@ page language="java" contentType="text/html" %>          — JSP element

<html>                                                       — template text
 <body bgcolor="white">

 <jsp:useBean                                                — JSP element
  id="userInfo"
  class="com.ora.jsp.beans.userinfo.UserInfoBean">
  <jsp:setProperty name="userInfo" property="*"/>
 </jsp:useBean>

  The following information was saved:                       — template text
  <ul>
   <li>User Name:

    <jsp:getProperty name="userInfo"                         — JSP element
     property="userName"/>

   <li>Email Address:                                        — template text

    <jsp:getProperty name="userInfo"                         — JSP element
    property="emailAddr"/>

  </ul>                                                      — template text
 </body>
</html>
```

JSP Processing

JSP pages can be processed using JSP container only. Steps to be followed while processing the request for JSP page:

1. Client makes a request for required JSP page to the server. The server must have JSP container so that JSP request can be processed.
2. On receiving the request, JSP container searches and reads the corresponding servlet. Every **template text** is translated into corresponding println statement. Every **JSP element** is converted into Java code. This phase is called **translation phase**. The output of this phase is a servlet.
3. The servlet is then compiled to generate the class file. Using this class the response can be generated. This phase is called **request processing phase**.
4. The JSP container thus executes the servlet class file.
5. A requested page is then returned to the client as **response.**

JSP Processing



Declarations

Scripting Elements:

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
<%  java source code %>
```

Example of JSP scriptlet tag

```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

Declarations

JSP expression tag

The code placed within JSP expression tag is written to the output stream of the response. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

```
<%=  statement %>
```

Example of JSP expression tag

```
<html>
    <body>
        <%= "welcome to jsp" %>
    </body>
</html>
```

Declarations

JSP Declaration Tag

The JSP declaration tag is used to declare fields and methods.

Syntax of JSP declaration tag

<%! field or method declaration %>

Example of JSP declaration tag that declares field

```
//data.jsp

<html>
        <body>
                <%! int data=50; %>
                <%= "Value of the variable is:"+data %>
        </body>
</html>
```

Declarations

Example of JSP declaration tag that declares method

```
//cube.jsp
<html>
        <body>
                <%!
                        int cube(int n){
                                return n*n*n*;
                        }
                %>
                <%= "Cube of 3 is:"+cube(3) %>
        </body>
</html>
```

Declarations

Example: MethodDemo.jsp

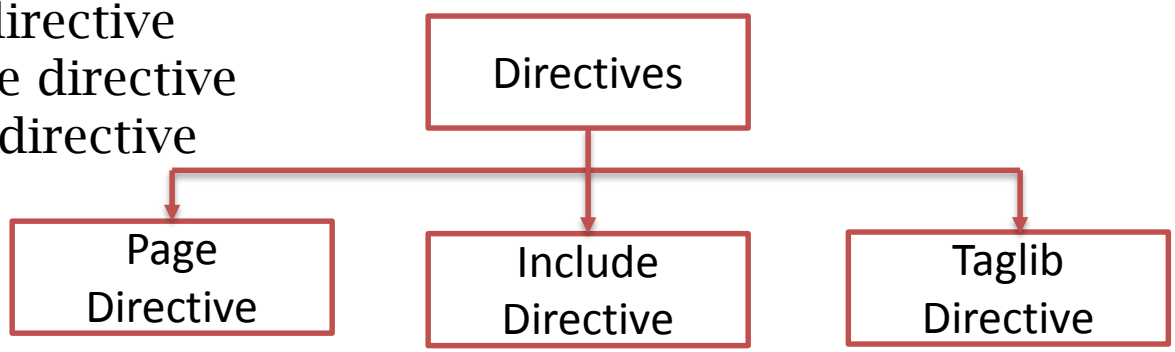
```
<%@ page language="java" contentType="text/html"%>
        <%!
                String msg="Hello";
        %>
        <%!public String MyFunction(String msg)
        {
                return msg;
        }
        %>
<html>
        <head>
                <title>Use of Method</title>
        </head>
                <body>
                        <%out.println("Before function call:"+msg);%>
                <br/>
                        After Function Call: <%=MyFunction("JSP")%>
                </body>
</html>
```

Directives

The jsp directives are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

- page directive
- include directive
- taglib directive



Syntax of JSP Directive

<%@ directive attribute="value" %>

Directives

JSP page directive

The page directive defines attributes that apply to an entire JSP page.

Syntax:
<%@ page attribute="value" %>

Attributes of JSP page directive

- import
- contentType
- extends
- info
- buffer
- language
- isELIgnored
- isThreadSafe
- autoFlush
- session
- pageEncoding
- errorPage
- isErrorPage

Example:
%@page import =“java.io.*”%
<%@page language=“java”%>
<%@page contentType=“text/html”%>

Directives

Jsp Include Directive

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

Advantage of Include directive

- Code Reusability

Syntax of include directive

<%@ include file="resourceName" %>

```
<html>
<body>

<%@ include file="header.html" %>
 Today is: <%= java.util.Calendar.getInstance().getT
ime() %>

</body>
</html>
```

Directives

JSP Taglib directive

The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag(created by user) section we will use this tag so it will be better to learn it in custom tag.

Syntax JSP Taglib directive

<%@ taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>

```
<html>
<body>

<%@ taglib uri="WEB-INF/mytags.tld" prefix="mytag" %>

<mytag:currentDate/>

</body>
</html>
```

Directives

MyTagHandler.java

```
import java.util.Calendar;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;
public class MyTagHandler extends TagSupport{

public int doStartTag() throws JspException {
    JspWriter out=pageContext.getOut();//returns the instance of JspWriter
    try{
     out.print(Calendar.getInstance().getTime());//printing date and time using JspWriter
    }catch(Exception e){System.out.println(e);}
    return SKIP_BODY;//will not evaluate the body content of the tag
}
}
```

Directives

mytags.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
        PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
    "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">

<taglib>

  <tlib-version>1.0</tlib-version>
  <jsp-version>1.2</jsp-version>
  <short-name>simple</short-name>
  <uri>http://tomcat.apache.org/example-taglib</uri>

<tag>
<name>today</name>
<tag-class>MyTagHandler</tag-class>
</tag>
</taglib>
```

Index.jsp

```
<%@ taglib uri="WEB-INF/mytags.tld" prefix="m" %>
Current Date and Time is: <m:today/>
```

Directives

```
<%@ page language="java" contentType="text/html" %>
<%@ page import="java.util.*" %>

<html>
<body>
        Todays Date is:<%= new Date().toString()%>

</body>
</html>
```

Expressions

The expression tag is used to represent the expression in JSP page.

Syntax:

```
<%= Java Expression %>
```

```
<html>
    <head>
        <title>JSP expression Demo </title>
    </head>
    <body>
        Value of Expression is:
        <%=(10*20)%>
    </body>
</html>
```

Code Snippets

The code that appears between <% and %> delimiters is called a **scriptlet.** Scriplets are nothing but java code enclosed within <% and %> tags.

Templatetext.jsp

```
<%@ page language="java" contentType="text/html"%>
        <html>
                <head>
                        <title>Demo for template text</title>
                </head>
        <body bgcolor="gray">
        <h1>Hello</h1>
        <h1>Good Morning</h1>
        <p>
                <% out.print("JSP is equal to HTML and JAVA");%>
        </p>
        </body>
        </html>
```

Code Snippets

Scriptletdemo.jsp

```
<%@ page language="java" contentType="text/html"%>
        <html>
                <head>
                        <Introduction to scriptlet</title>
                </head>
        <body>
                <strong>
        <%out.println("Value of I is:");%>
        <%fot(int i=0;i<5;i++){
                out.print("<br/>");
                if(i%2==0)
                        out.println("Even value is:"+i);
                else
                        out.println("Odd value is:"+i);
                }%>
                </strong>
        </body>
</html>
```

Implicit Objects

The implicit objects are predefined variables used to access request and application data. These objects are used by the scripting elements.

A list of the 9 implicit objects is given below:

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	PageContext
page	Object
exception	Throwable

Implicit Objects

1. JSP out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

PrintWriter out=response.getWriter();

```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
</body>
</html>
```

Implicit Objects

2. JSP request implicit object

The JSP request is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container. It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

index.html

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

welcome.jsp

```
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
```

Implicit Objects

3. JSP response implicit object

In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request.

It can be used to add or manipulate response such as redirect response to another resource, send error etc.

index.html

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

welcome.jsp

```
<%
response.sendRedirect("http://www.google.com");
%>
```

Implicit Objects

4. JSP config implicit object

In JSP, config is an implicit object of type ServletConfig. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page. using Config we can get initialization parameters of an individual servlet mapping.

index.html

```
<form action="welcome">
        <input type="text" name="uname">
        <input type="submit" value="go"><br/>
</form>
```

web.xml file

```
<web-app>

<servlet>
<servlet-name>Hello</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
</servlet>


<init-param>
        <param-name>dname</param-name>
        <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</init-param>


<servlet-mapping>
        <servlet-name>Hello</servlet-name>
        <url-pattern>/welcome</url-pattern>
```

welcome.jsp

```
<%
out.print("Welcome "+request.getParameter("uname"));

String driver=config.getInitParameter("dname");
out.print("driver name is="+driver);
%>
```

Implicit Objects

5. JSP application implicit object

In JSP, application is an implicit object of type *ServletContext*.
The instance of ServletContext is created only once by the web container when application or project is deployed on the server.
This object can be used to get initialization parameter from configuaration file (web.xml).
It can also be used to get, set or remove attribute from the application scope.
This initialization parameter can be used by all jsp pages.

index.html

```
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

Implicit Objects

web.xml file

```
<web-app>

<servlet>
<servlet-name>sonoojaiswal</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
</servlet>

<servlet-mapping>
<servlet-name>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>

</web-app>
```

welcome.jsp

```
<%

out.print("Welcome  "+request.getParameter("uname"));


String driver=application.getInitParameter("dname");
out.print("driver name is="+driver);

%>
```

Implicit Objects

6. session implicit object

In JSP, session is an implicit object of type HttpSession.The Java developer can use this object to set,get or remove attribute or to get session information.

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

welcome.jsp

```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

session.setAttribute("user",name);

<a href="second.jsp">second jsp page</a>

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

second.jsp

```
<html>
<body>
<%

String name=(String)session.getAttribute("user");
out.print("Hello "+name);

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

Implicit Objects

7. pageContext implicit object

In JSP, pageContext is an implicit object of type PageContext class.The pageContext object can be used to set,get or remove attribute from one of the following scopes:
page
request
session
application
In JSP, page scope is the default scope.

index.html

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

Implicit Objects

welcome.jsp

```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

pageContext.setAttribute("user",name,PageContext.SESSION_SCOPE);

<a href="second.jsp">second jsp page</a>

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

second.jsp

```
<html>
<body>
<%

String name=(String)pageContext.getAttribute("user",PageContext.SESSION_SCOPE);
out.print("Hello "+name);

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

Implicit Objects

8. Page implicit object:

In JSP, page is an implicit object of type Object class.This object is assigned to the reference of auto generated servlet class. It is written as:

Object page=this;

For using this object it must be cast to Servlet type.For example:

<% (HttpServlet)page.log("message"); %>

Since, it is of type Object it is less used because you can use this object directly in jsp.For example:

<% this.log("message"); %>

Implicit Objects

9. exception implicit object:

In JSP, exception is an implicit object of type java.lang.Throwable class. This object can be used to print the exception. But it can only be used in error pages.It is better to learn it after page directive. Let's see a simple example:

error.jsp

```
<%@ page isErrorPage="true" %>
<html>
<body>

Sorry following exception occured:<%= exception %>

</body>
</html>
```

Implicit Objects

Index.html

```
<html>
<head>
<title>Enter two Integers for Division</title>
</head>
<body>
<form action="division.jsp">
Input First Integer:<input type="text" name="firstnum" />
Input Second Integer:<input type="text" name="secondnum" />
<input type="submit" value="Get Results"/>
</form>
</body>
</html>
```

division.jsp

```
<%@ page errorPage="exception.jsp" %>
<%
String num1=request.getParameter("firstnum");
String num2=request.getParameter("secondnum");
int v1= Integer.parseInt(num1);
int v2= Integer.parseInt(num2);
int res= v1/v2;
out.print("Output is: "+ res);
```

exception.jsp

```
<%@ page isErrorPage="true" %>
Got this Exception: <%= exception %>
Please correct the input data.
```

Using Beans in JSP Pages

Java beans are reusable components. We can use simple Java bean in the JSP. This helps us in keeping the business logic separate from the presentation logic. Beans are used in the JSP pages as the instance of class. We must specify the scope of the bean for its existence in JSP. When the bean is present in particular scope its id is also available in that scope.

There are various scopes using which the bean can be used in JSP page.

1. **Page scope:** The bean object gets disappeared as soon as the current page gets discarded. The default scope for a bean in JSP page is a page scope.
2. **Request scope:** The bean object remains in existence as long as the request object is present.
3. **Session scope:** A session can be defined as a specific period of time the user spends browsing the site.
4. **Application scope:** During application scope the bean will get stored to ServletContext. Hence particular bean is available to all the servlets in the same web application.

Using Beans in JSP Pages

A Java Bean is a Java Class that meets the following requirements:

- It has a public, no argument constructor.
- It implements the java.io.Serializable or java.io.Externalizable interface.
- Its properties are accessible using methods that are written following a standard naming convention.

Example: (Compile & save . class file into C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps\123\WEB-INF\classes)

```
package bean;
public class Factorial implements java.io.Serializable
{
        int n;
        public int getValue()
        {
                int prod=1;
                for(int i=2;i<=n;i++)
                        prod*=i;
                return prod;
        }
        public void setValue(int v){ n=v;}
}
```

Using Beans in JSP Pages

useBean

A JSP action element <jsp:useBean> instantiates a Java Bean object into the JSP page.

<jsp:useBean id="object_name" class="class_name"
scope="page|request|session|application"/>

id- name of the object to be created.
<% class_name object_name=new class_name();%>

Ex:
<jsp:useBean id="fact" scope="page" class="bean.Factorial"/>
equivalent to
<% bean.Factorial fact=new bean.Factorial();%>

Using Beans in JSP Pages

setProperty

The <jsp:setProperty> action tag assigns a new value to the specified property of the specified bean object.

<jsp:setProperty name="obj_name" property="prop_name" value="prop_value"/>

The object name, property name, and its values are specified by the name, property and value attributes. Equivalent to calling setProp_name() method on the specified object obj_name as follows:

<% obj_name.setProp_name(prop_value);%>

To set a property of our bean object fact, use

<jsp:setProperty name="fact" property="value" value="5"/>

The equivalent scriptlet is <% fact.setValue(5); %>

Using Beans in JSP Pages

getProperty

The <jsp:getProperty> action tag retrieves the value of specified property of the specified bean object. the value is converted to a string.

<jsp:getProperty name="obj_name" property="prop_name"/>

The object name, property name are specified by the name, property attributes.
Equivalent to calling getProp_name() method on the specified object obj_name as follows:

<% obj_name.getProp_name(); %>

To get a value of our bean object fact, use

<jsp:getProperty name="fact" property="value" />

The equivalent scriptlet is <% fact.getValue(); %>

Using Beans in JSP Pages

Factorial.jsp

```
<table border="1">
        <caption>Factorial table</caption>
        <tr>
                <th width="50">n</th>
                <th width="100">n!</th>
        </tr>
        <jsp:useBean id="fact" scope="page" class="bean.Factorial"/>
        <%
                for(int i=2;i<6;i++)
                {
        %>
        <jsp:setProperty name="fact" property="value" value="<%=i%>"/>

        <tr>
                <td><%=i%></td>
                <td><jsp:getProperty name="fact" property="value" /></td>
        </tr>
        <%
        }
        %>
</table>
```

Using Beans in JSP Pages

Other Usage:

Once a bean object is loaded into the page, it can be used exactly like other object in scripting elements in the same JSP page.

```
<jsp:useBean id="fact" scope="page" class="bean.Factorial"/>
<% fact.setValue(6);%>
<%=fact.getValue()%>
```

Using Cookies

- Cookies are small text files that are stored in the clients machine.
- Used to keep track of the users who browse the web.
- The browser stores information on the local machine and makes use of this information next time when the user is browsing the web.

Various operations in handling cookies are-

1. Create Cookie
2. Read Cookie
3. Delete Cookie

Create Cookie:

Step1: Cookie is created using Cookie class. It requires two parameters name and value.
Ex: Cookie c=new Cookie("name","value");

Step2: Set the validity period using setMaxAge.
Ex: c.setMaxAge(60*60*24);

Step3: Add cookie in HTTP response header as
response.addCookie(c);

Using Cookies

Read Cookie:

Step1: Cookie is retrieved using getCookies() method.
Ex: Cookie[] cookies=request.getCookies();

Step2: Use getNAme() and getValue() methods to read cookies.

Delete Cookie:

Step1: Read already created cookie and store it in cookie object.
Ex: Cookie c=new Cookie("name","");

Step2: Set period of existence as 0 by setMaxAge method.
c.setMaxAge(0);

Step3: Add this cookie back to response header.
response.addCookie(c);

Using Cookies

input.html

```
<html>
<body>
<form method="post"
action="http://localhost:8080/123/Createcookie.jsp">
        Username:<input type="text" name="name">
        City:<input type="text" name="city">
        <input type="submit" value="SUBMIT">
        </form>
</body>
</html>
```

Createcookie.jsp

```
<%@page language="java" import="java.util.*"%>
<%
        String name=request.getParameter("name");
        String city=request.getParameter("city");

        Cookie namecookie=new Cookie("name",name);
        Cookie citycookie=new Cookie("city",city);

        response.addCookie(namecookie);
        response.addCookie(citycookie);

        namecookie.setMaxAge(60*60*24);
        citycookie.setMaxAge(60*60*24);
%>
<h3>
<a href="http://localhost:8080/123/Readcookie.jsp">Click here to c
```

Using Cookies

Readcookie.jsp

```
<h3>Reading the Cookie</h3>
<%
Cookie[] cookies=request.getCookies();
for(int i=0;i<cookies.length;i++)
{
        out.println("Cookie Name:"+cookies[i].getName()+"<br>");
        out.println("Cookie Value:"+cookies[i].getValue()+"<br>");
}
%>
<h3>
<a href="http://localhost:8080/123/Deletecookie.jsp">Click here to Delete
cookie..</a>
</h3>
```

Using Cookies

Deletecookie.jsp

```
<%
Cookie namecookie=new Cookie("name","");
namecookie.setMaxAge(0);
namecookie.setValue("");
response.addCookie(namecookie);

Cookie citycookie=new Cookie("city","");
citycookie.setMaxAge(0);
citycookie.setValue("");
response.addCookie(citycookie);
%>

<h3>
<a href="http://localhost:8080/123/Readcookie.jsp">Click here to chech
the deletion..</a>
</h3>
```

Session Handling

There are some web applications in which user moves from one page to another. Then it becomes necessary to keep track of user data. That means user data must be consistent throughout the web applications. To bring this consistency of user data between the web pages normally there is a use of an implicit object called session. Using session we can save the data of particular page.

First.jsp

```
<%@ page language="java"%>
<html>
        <body>
                <form method="post"
action="http://localhost:8080/123/Second.jsp">
                Username:<input type="text" name="UserName">
                Pasword:<input type="password" name="Password">
                <input type="submit" value="ENTER">
                </form>
        </body>
</html>
```

Session Handling

Second.jsp

```
<%@ page language="java"%>
<%
        String username=request.getParameter("UserName");
        String pwd=request.getParameter("Password");
        session.setAttribute("Username",username);
        session.setAttribute("Password",pwd);
%>
<html>
        <body>
                <h3>
        <a href="http://localhost:8080/123/Third.jsp">Click here to vie next
session</a>
                </h3>

</html>
```

Session Handling

Third.jsp

```
<%@ page language="java"%>
<%
        String username=(String)session.getAttribute("Username");
        String password=(String)session.getAttribute("Password");
%>

<html>
        <body>
                <h3>
        User Name: <%=username%>
        Password:<%=password%>
                </h3>

</html>
```