JSP

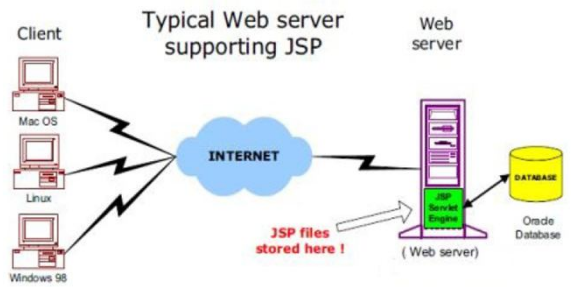
Mrs.V.Mareeswari
Assistant Professsor
School of Information Technology and Engineering
VIT University

Cabin No:SJT 210-A30

Java Server Pages

- JSP for short is Sun's solution for developing dynamic web sites. JSP provide excellent server side scripting support for creating database driven web applications. JSP enable the developers to directly insert java code into jsp file, this makes the development process very simple and its maintenance also becomes very easy. JSP pages are efficient, it loads into the web servers memory on receiving the request very first time and the subsequent calls are served within a very short period of time.
- Java is known for its characteristic of "write once, run anywhere." JSP pages are platform independent. Your port your .jsp pages to any platform. The advantage of JSP is that they are document-centric.
- **JavaServer Pages** (**JSP**) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Latest Release : JSP 2.2

- In today's environment most web sites servers dynamic pages based on user request. Database is very convenient way to store the data of users and other things. JDBC provide excellent database connectivity in heterogeneous database environment. Using JSP and JDBC its very easy to develop database driven web application.



Advantage of JSP over Servlet

1) Extension to Servlet : JSP technology is the extension to servlet technology. We can use all the features of servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

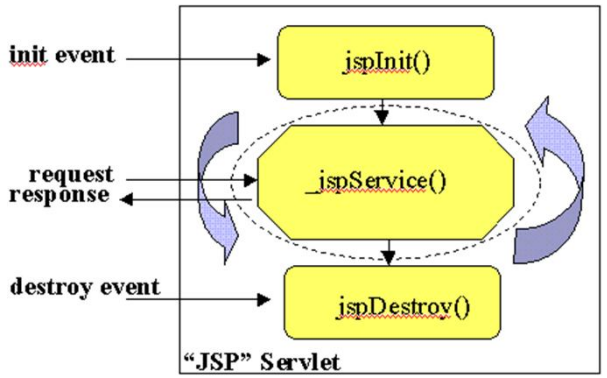
2) Easy to maintain : JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

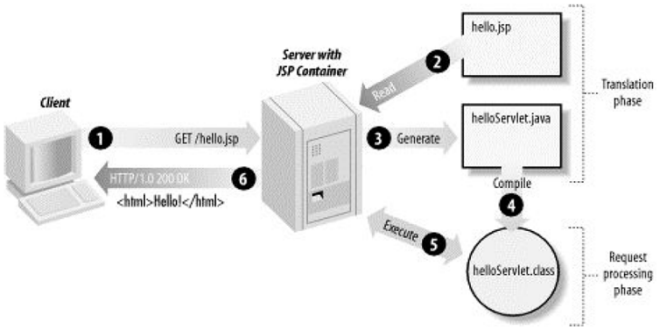
4) Less code than Servlet : In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces the code. Moreover, we can use EL, implicit objects etc.

Life Cycle of JSP



5 V.MAREESWARI / AP / SITE 26 September 2014

HTML Form Processing Using JSP



6 V.MAREESWARI / AP / SITE 26 September 2014

Cont...

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with .jsp instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println() statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.

7 V.MAREESWARI / AP / SITE 26 September 2014

Cont...

- The JSP engine compiles the servlet into an executable class and forwards the original request to a servletengine.
- A part of the web server called the servletengine loads the Servletclass and executes it. During execution, the servlet produces an output in HTML format, which the servletengine passes to the web server inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

8 V.MAREESWARI / AP / SITE 26 September 2014

Cont..

- This translation and compilation phase occurs only when the JSP file is requested for the first time, or if it undergoes any changes to the extent of getting retranslated and recompiled. For each additional request of the JSP page thereafter, the request directly goes to the servlet byte code, which is already in memory. Thus when a request comes for a servlet, an init() method is called when the Servlet is first loaded into the virtual machine, to perform any global initialization that every request of the servlet will need. Then the individual requests are sent to a service() method, where the response is put together. The servlet creates a new thread to run service() method for each request. The request from the browser is converted into a Java object of type HttpServletRequest, which is passed to the Servlet along with an HttpServletResponse object that is used to send the response back to the browser. The servlet code performs the operations specified by the JSP elements in the .jsp file.

JSP Tags

- **Directives**
  In the directives we can import packages, define error handling pages or the session information of the JSP page.
- **Declarations**
  This tag is used for defining the functions and variables to be used in the JSP.
- **Scriplets**
  In this tag we can insert any amount of valid java code and these codes are placed in *_jspService* method by the JSP engine.
- **Expressions**
  We can use this tag to output any data on the generated page. These data are automatically converted to string and printed on the output stream.

Directives

Syntax of JSP directives is:

```
<%@directive attribute="value" %>
```

Where directive may be:

- page: page is used to provide the information about it.
  Example: <%@page language="java" %>
- include: include is used to include a file in the JSP page.
  Example: <%@ include file="/header.jsp" %>
- taglib: taglib is used to use the custom tags in the JSP pages (custom tags allows us to defined our own tags).
  Example: <%@ taglib uri="tlds/taglib.tld" prefix="mytag" %>

and attribute may be:

- language="java“
  This tells the server that the page is using the java language. Current JSP specification supports only java language.
- session="true“
  When this value is true session data is available to the JSP page otherwise not. By default this value is true.
- errorPage="error.jsp"
  errorPage is used to handle the un-handled exceptions in the page.
- contentType="text/html;charset=ISO-8859-1"
  Use this attribute to set the mime type and character set of the JSP.

```
<%@page language="java" session="true" contentType="text/html;
charset=ISO-8859-1"  errorPage="error.jsp" %>
```

Declarative

- Syntax of JSP Declaratives are:

```
<%!
      //java codes
 %>
```

- Variables and functions defined in the declaratives are class level and can be used anywhere in the JSP page.

Example Program : Count Display

```
<%@page
  contentType="text/html" %>
<html><body>
<%!
int cnt=0;
private int getCount()
{
  cnt++;
  return cnt;
}
%>
<p>Values of Cnt are:</p>
<p><%=getCount()%></p>
<p><%=getCount()%></p>
<p><%=getCount()%></p>
<p><%=getCount()%></p>
<p><%=getCount()%></p>
<p><%=getCount()%></p>
</body></html>
```


Scriptles / Implicit Objects

- Syntax of JSP Scriptles are: `<% //java codes %>`
- To simplify code in JSP expressions and scriptlets, you are supplied with nine automatically defined variables, sometimes called ***implicit objects*.**

1. request: request represents the clients request and is a subclass of **HttpServletRequest**. Use this variable to retrieve the data submitted along the request.

```
<%
String userName=null;
userName=request.getParameter("userName");   %>
```


2. response: This is the **HttpServletResponse** associated with the response to the client.

```
<%   //Set refresh, autoload time as 5 seconds
response.setIntHeader("Refresh", 5); %>
```

3. out: This is the **PrintWriter** object used to send output to the client.

```
<% out.println("WELCOME"); %>
```

4. session: session represents the **HTTP session object** associated with the request.

```
<% out.print( session.getId()); %>
```

5. application : This is the **ServletContextobject** associated with application context.

```
<% application.setAttribute("hitCounter", hitsCount); %>
```

6. config:This is the **ServletConfigobject** associated with the page.

<% config.getServletName(); %>

7. pageContext: The pageContext object is an instance of a **javax.servlet.jsp.PageContext object**. The pageContext object is used to represent the entire JSP page. This object is intended as a means to access information about the page while avoiding most of the implementation details.

8. page: This is simply a synonym for **this**, and is used to call the methods defined by the translated servlet class.

<%@ page import="javax.servlet.http.*" %>

<%@ page errorPage="ShowError.jsp" %>

9.Exception: The **Exception** object allows the exception data to be accessed by designated JSP.

Expressions

- JSP Expressions start with <%= and ends with %>. Between these this you can put anything and that will converted to the String and that will be displayed.
- Example:

<%="HelloWorld!" %>→ display 'HelloWorld!'

<%=getCount()%> → return value of getCount() method

<%= (new java.util.Date())%> → Date

<%= "Welcome "+request.getParameter("uname") %> → FormValue

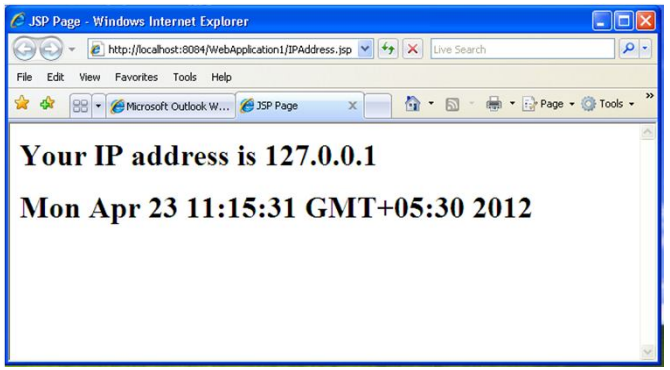
JSP Comments:

<%-- This is JSP comment --%>

Simple Program

```
<html>
  <head>
        <title>JSP Page</title>
  </head>
  <body>
    <h1>
<% out.println("Your IP address is " + request.getRemoteAddr()); %>
</h1>
    <h1><%= (new java.util.Date())%></h1>
  </body>
</html>
```


Output



ExceptionExample.jsp → Arithmetic.jsp→ if error →error.jsp

```
<form action="http://localhost:8084/Project/Arithmetic.jsp"
method="post">
   Number1:<input type="text" name="no1"/><br/>
   Number2:<input type="text" name="no2"/><br/>
   <input type="submit" value="Division"/></form>
```

```
<%@ page errorPage="error.jsp" %>
<%      String num1=request.getParameter("no1");
   String num2=request.getParameter("no2");
   int a=Integer.parseInt(num1);    int b=Integer.parseInt(num2);
   int c=a/b;
   out.print("division of numbers is: "+c);   %>
```

```
<%@ page isErrorPage="true" %>
<h3>Sorry an exception occured!</h3>
 Exception is: <%= exception %>
```

21 V.MAREESWARI / AP / SITE 26 September 2014

JSP Action Tags

The action tags are used to control the flow between pages and to use Java Bean.

jsp:forward	forwards the request and response to another resource.
jsp:include	includes another resource.
jsp:useBean	creates or locates bean object.
jsp:setProperty	sets the value of property in bean object.
jsp:getProperty	prints the value of property of the bean.
jsp:plugin	embeds another components such as applet.
jsp:param	sets the parameter value. It is used in forward and include mostly.
jsp:fallback	can be used to print the message if plugin is working. It is used in jsp:plugin.

22 V.MAREESWARI / AP / SITE 26 September 2014

jsp:forward

Index.jsp

```
<html> <body>
<h2>this is index page</h2>
<jsp:forward page="printdate.jsp" />
</body> </html>
```

Printdate.jsp

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

23 V.MAREESWARI / AP / SITE 26 September 2014

jsp:include

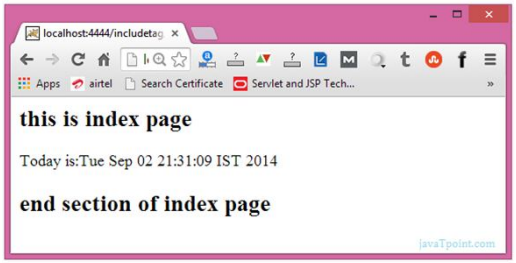
- The **jsp:include action tag** is used to include the content of another resource it may be jsp, html or servlet.
- **Advantage is code reusability** : We can use a page many times such as including header and footer pages in all pages. So it saves a lot of time.

JSP include directive	JSP include action
includes resource at translation time.	includes resource at request time.
better for static pages.	better for dynamic pages.
includes the original content in the generated servlet.	calls the include method.

24 V.MAREESWARI / AP / SITE 26 September 2014

Index.jsp

```
<h2>this is index page</h2>
 <jsp:include page="printdate.jsp" />
<h2>end section of index page</h2> s
```



Java Bean

A Java Bean is a java class that should follow following conventions:

- It should have a no-argument constructor.
- It should be Serializable.
- It should provide methods to set and get the values of the properties, known as getter and setter methods.

Why use Java Bean?

- According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object, so we can access this object from multiple places. Moreover, it provides the easy maintenance.

Accessing Bean

```
<jsp:useBean id="date" class="java.util.Date" />
<p>The date/time is <%= date %>
```

```
 <jsp:useBean
id= "instanceName"
scope= "page | request | session | application"
class= "packageName.className"
type= "packageName.className"
beanName="packageName.className |
<%= expression >" >
</jsp:useBean>
```


Attributes and Usage of jsp:useBean action tag

- **id:** is used to identify the bean in the specified scope.
- **scope:** represents the scope of the bean. It may be page, request, session or application. The default scope is page.
  - **page:** specifies that you can use this bean within the JSP page. The default scope is page.
  - **request:** specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.
  - **session:** specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.
  - **application:** specifies that you can use this bean from any JSP page in the same application. It has wider scope than session.

Cont..

- **class:** instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract.
- **type:** provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated.
- **beanName:** instantiates the bean using the java.beans.Beans.instantiate() method.

29 V.MAREESWARI / AP / SITE 26 September 2014

Bean Creation

```
package com.tutorialspoint;
public class StudentsBean implements java.io.Serializable
{
  private String firstName = null;
  private String lastName = null;
  private int age = 0;
  public StudentsBean() {  }
  public String getFirstName(){     return firstName;   }
  public String getLastName(){     return lastName;   }
  public int getAge(){     return age;   }
```

30 V.MAREESWARI / AP / SITE 26 September 2014

```
public void setFirstName(String firstName){
    this.firstName = firstName;   }
  public void setLastName(String lastName){
    this.lastName = lastName;   }
  public void setAge(Integer age){
    this.age = age;
  }
}
```

31 V.MAREESWARI / AP / SITE 26 September 2014

Accessing Bean

```
<html> <head>
<title>get and set properties Example</title> </head> <body>
<jsp:useBean id="students"
            class="com.tutorialspoint.StudentsBean">
  <jsp:setProperty name="students" property="firstName"
            value="Zara"/>
  <jsp:setProperty name="students" property="lastName"
            value="Ali"/>
  <jsp:setProperty name="students" property="age"
            value="10"/>
</jsp:useBean>
```

32 V.MAREESWARI / AP / SITE 26 September 2014

```
<p>Student First Name:
  <jsp:getProperty name="students" property="firstName" />
</p>
<p>Student Last Name:
  <jsp:getProperty name="students" property="lastName" />
</p>
<p>Student Age:
  <jsp:getProperty name="students" property="age" />
</p>
</body> </html>
```


jsp:plugin

```
<jsp: plugin type="applet" code="MyApplet.class" codebase=" /">
        <jsp:params>
                <jsp:param name="myParam" value="marees" />
        </jsp:params>
        <jsp:fallback>
                <b>Unable to load applet</b>
        </jsp:fallback>
</jsp:plugin>
```


JSP + MS Access + JDBC

JSPDatabaseLogin.jsp

```
 <form
action="http://localhost:8084/Project/JSPDatabaseValidation.jsp"
method="post">
    Name:<input type="text" name="name" /><br />
    Password:<input type="password" name="pwd" /><br />
    <input type="submit" value="login" />
</form>
```


JSPDatabaseValidation.jsp

```
<%@ page import = "java.sql.*" %>
<%
String name=request.getParameter("name");
String password=request.getParameter("pwd");
int verified=0;
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con =
DriverManager.getConnection("jdbc:odbc:DSN_Account");
//university.mdb
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from students");
```

```
while(rs.next())
{
        String dbname=rs.getString("Name");
        String dbpassword=rs.getString("Password");
        if(name.equals(dbname) && password.equals(dbpassword))
                verified=1;
}
if(verified==1)
    out.println("Valid User");
else
    out.println("InValid User.Login Again");
%>
```

37 V.MAREESWARI / AP / SITE 26 September 2014