Java Server Pages

What is JSP?

Java Server Pages (JSP) is a technology that lets you mix regular, static HTML with dynamically-generated HTML. Many Web pages that are built by CGI programs are mostly static, with the dynamic part limited to a few small locations. But most CGI variations, including servlets, make you generate the entire page via your program, even though most of it is always the same. JSP lets you create the two parts separately.

What Is a JSP Page?

- A JSP page is a text document that contains two types of text: static data, which can be expressed in any text-based format (such as HTML, SVG, WML, and XML), and JSP elements, which construct dynamic content.
- JavaServer Pages (JSP) is a technology for developing web pages that support dynamic content which helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
- A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.
- Using JSP, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages and sharing information between requests, pages etc.
- The recommended file extension for the source file of a JSP page is .jsp. The page can be composed of a top file that includes other files that contain either a complete JSP page or a fragment of a JSP page. The recommended extension for the source file of a fragment of a JSP page is .jspf.
- The JSP elements in a JSP page can be expressed in two syntaxes, standard and XML, though any given file can use only one syntax. A JSP page in XML syntax is an XML document and can be manipulated by tools and APIs for XML documents.

Why JSP?

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.
- JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.

- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.
- Finally, JSP is an integral part of J2EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding

Advantages of JSP over its counterparts?

- **vs. Active Server Pages (ASP).** ASP is a similar technology from Microsoft. The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS-specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.
- **vs. Pure Servlets.** JSP doesn't give you anything that you couldn't in principle do with a servlet. But it is more convenient to write (and to modify!) regular HTML than to have a zillion println statements that generate the HTML. Plus, by separating the look from the content you can put different people on different tasks: your Web page design experts can build the HTML, leaving places for your servlet programmers to insert the dynamic content.
- **vs. Server-Side Includes (SSI).** SSI is a widely-supported technology for including externally-defined pieces into a static Web page. JSP is better because it lets you use servlets instead of a separate program to generate that dynamic part. Besides, SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.
- **vs. JavaScript.** JavaScript can generate HTML dynamically on the client. This is a useful capability, but only handles situations where the dynamic information is based on the client's environment. With the exception of cookies, HTTP and form submission data is not available to JavaScript. And, since it runs on the client, JavaScript can't access server-side resources like databases, catalogs, pricing information, and the like.
- **vs. Static HTML.** Regular HTML, of course, cannot contain dynamic information. JSP is so easy and convenient that it is quite feasible to augment HTML pages that only benefit marginally by the insertion of small amounts of dynamic data. Previously, the cost of using dynamic data would preclude its use in all but the most valuable instances.

Requirement of Web Server processing and running JSP

The web server needs a JSP engine ie. container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages.

The following steps explain how the web server creates the web page using JSP:

- 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.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine 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.

Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet

Life Cycle of JSP
A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.
The following are the paths followed by a JSP

- Compilation
- Initialization
- Execution
- Cleanup

The three major phases of JSP life cycle are very similar to Servlet Life Cycle and they are as follows:

1.JSP Compilation:
When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.
The compilation process involves three steps:

1. Parsing the JSP.
2. Turning the JSP into a servlet.
3. Compiling the servlet.

2. JSP Initialization:
When a container loads a JSP it invokes the jspInit() method before servicing any requests. If you need to perform JSP-specific initialization, override the jspInit() method:
Typically initialization is performed only once and as with the servlet init method, you generally initialize database connections, open files, and create lookup tables in the jspInit method.

3. JSP Execution:
This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.Whenever a browser requests a JSP and the page has been loaded and initialized, the JSP engine invokes the **_jspService()** method in the JSP.The _jspService() method takes an **HttpServletRequest** and an **HttpServletResponse** as its parameters.
The _jspService() method of a JSP is invoked once per a request and is responsible for generating the response for that request and this method is also responsible for generating responses to all seven of the HTTP methods ie. GET, POST, DELETE etc.

4.JSP Cleanup:
The destruction phase of the JSP life cycle represents when a JSP is being removed from use by a container.The **jspDestroy()** method is the JSP equivalent of the destroy method for servlets. Override jspDestroy when you need to perform any cleanup, such as releasing database connections or closing open files.The jspDestroy () method has the following form:

JSP Life Cycle in simple terms
JSP life cycle can be divided into four phases: Translation, Initialization, execution and finalization.

Translation
In the translation phase, JSP engine checks for the JSP syntax and translate JSP page into its page implementation class if the syntax is correct. This class is actually a standard Java servlet. After that, JSP engine compiles the source file into class file and ready for use.
If the container receives the request, it checks for the changes of the JSP page since it was last translated. If no changes were made, It just loads the servlet otherwise the process of check, translate and compile occurs again. Because the compilation process takes time so JSP engine wants to minimize it to increase the performance of the page processing.

Initialization
After the translation phase, JSP engine loads the class file and create an instance of the servlet to handle processing of the initial request. JSP engines will call a method jspInit () to initialize the servlet. jspInit method is generated during the translation phase which is normally used for initializing application-level parameters and resources. You can also overide this method by using declaration.

Execution

After the initialization phase, the web container calls the method _jspService() to handle the request and returning a response to the client. Each request is handled is a separated thread. Be noted that all the scriptlets and expressions end up inside this method. The JSP directives and declaration are applied to the entire page so the are outside of this method.

Finalization

In the finalization phase, the web container calls the method jspDestroy(). This method is used to clean up memory and resources. Like jspInit () method, you can override the jspDestroy () method also to do your all clean up such as release the resources you loaded in the initialization phase.

JSP Language

JSP Language is based on Java Servlets i.e Java codes are embedded within HTML.The heart of JSP is JAVA.Various tags are present in JSP to enclose these java codes.It is discussed as the next topic.

JSP Tags

The tags in JSP are

The Scriptlet:

A scriptlet can contain any number of JAVA language statements, variable or method declarations, or expressions that are valid in the page scripting language.
Following is the syntax of Scriptlet:

```
<% // java code   %>
```

Sample Code

```
<html><head><title>JSP Page with scriptlet
tag</title></head>
<body>
<%
      out.println ("Hello! Welcome to the world of JSP");
%>
</body>
</html>
```

JSP Expression:

A JSP expression element contains a scripting language expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.Because the value of an expression is converted to a String, you can use an expression within a line of text, whether or not it is tagged with HTML, in a JSP file.The expression element can contain any expression that is valid according to the Java Language Specification but you cannot use a semicolon to end an expression.

Syntax of JSP Expression:

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

Sample Code

```
<html><head>
<title>JSP Page with scriptlet tag</title>
</head>
<body>
<%
    int a = 5; int b = 6;
    int c = a + b;
%>
<h1> The Result is : <%= c %> </h1>
</body>

</html>
```

The expression tag needn't be embedded within scriptlet tag.It is not an error if it is embedded with the scriplet tag.

JSP Comments:

JSP comment marks text or statements that the JSP container should ignore. A JSP comment is useful when you want to hide or "comment out" part of your JSP page.
Following is the syntax of JSP comments:

```
<%-- Any String --%>
```

Sample Code

```
<html><head>
<title>JSP Page with scriptlet tag</title>
</head>
<body>
<%
    int a = 5; int b = 6;
    int c = a + b;
%>
<%-- The JSP to find sum of two numbers --%>
<h1> The Result is : <%= c %> </h1>
```

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

JSP Directives:

A JSP directive affects the overall structure of the servlet class. It usually has the following form:

Syntax:

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

The various directives are

Directive	Description
<%@ page ... %>	Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
<%@ include ... %>	Includes a file during the translation phase.
<%@ taglib ... %>	Declares a tag library, containing custom actions, used in the page

Sample code:

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<html><head>
<title>JSP Page with scriptlet tag</title>
</head>
<body>
<%= new Date()%>
</body>
</html>
```

JSP Declarations:

A declaration declares one or more variables or methods that you can use in Java code later in the JSP file. You must declare the variable or method before you use it in the JSP file.

Syntax:

```
<%! Java Declarations %>
```

Sample Code

```
<html><head>
<title>JSP Page with scriptlet tag</title>
</head>
<body>

<%!
   int a = 5;
   int b = 6;
   int c;
%>
<% c = a + b; %>
<%-- The JSP to find sum of two numbers --%>
<h1> The Result is : <%= c %> </h1>
</body>
</html>
```

Control Statements:

JSP is java based,so all the control statement available in java can be used in JSP.The java code should be enclosed within scriptlet tag.

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

    <h1>JSP Page</h1>

   <%! int a = 5;
       int b = 6;
       int arr[ ]={ 1,2,3,4,5,6};
       int ser;
       boolean flag = false;
   %>

   <% if ( a > b )
   %>
   <b>a is greater than b</b>
```

```
    <%
       for(int i = 0 ; i < arr.length ; i++){
          if ( arr[ i ] == ser){
            flag = true;
            break;
          }
       }
    if ( flag){%>
       <b> Found </b>

    <% } else {%>
        <b>not found</b>
    <%}%>

    </body>
</html>
```

JSP with database

Program 1: Database Connectivity

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import ="java.sql.*"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP-Database</title>
    </head>
    <body>

    <h1>JSP-Database</h1>

   <%
       Connection conn;
       Statement st;
       ResultSet res;
       try{

        Class.forName("com.mysql.jdbc.Driver");

conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/e",
```

```
"root","amirtha");
           st=conn.createStatement();
         res=st.executeQuery("select * from employee2");
   int i=0;%>
           <table border = "1" >
              <tr>
              <td>num</td>
              <td>fac code</td>
              </tr>
           <%while(res.next())

           {%>

              <tr>
              <td><b><%=res.getInt(1)%> </td>
              <td><b><%=res.getString(2)%></b></td>
              </tr>


            <% }%>
           </table>
         <%     st.close();
                 conn.close();
            }catch (ClassNotFoundException err) { }
            catch (SQLException err) {  }


   %>

    </body>
</html>
```

Program 2: Database Connectivity(simple version)

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import ="java.sql.*"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
```

```
    <h1>JSP Page</h1>
    <%
       Connection conn;
       Statement st;
       ResultSet res;
       try{

        Class.forName("com.mysql.jdbc.Driver");

conn=DriverManager.getConnection("jdbc:mysql://localhost:33
06/e","root","amirtha");
           st=conn.createStatement();
         res=st.executeQuery("select * from employee2");
         int i=0;
           while(res.next())
         {

              out.println(res.getInt(1));
              out.println( res.getString(2));


       }

       st.close();
                 conn.close();
}catch (ClassNotFoundException err) { }
              catch (SQLException err) {  }


   %>


    </body>
</html>
```

Program 4: FORM Data-jhtml.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">

<html>
  <head>
    <title></title>
  </head>
  <body>
  <form method ="get" action ="jsp2.jsp">
```

```
      <input type ="text" name="user"/>
      <input type="submit" name="submit"/>
  </form>
  </form>
  </body>
</html>
```

jsp2.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
    <h1>JSP Page</h1>
   <%= request.getParameter("user")%>
    </body>
</html>
```

Program 5: Sessions in JSP

Logo.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
        <head>
                <title>Sessions Example</title>
        </head>
        <body bgcolor="Aqua">
        <form method = "post" action = "loginPro.jsp">
            <h1 style="color:blue"><center>LOGIN
PAGE</center></h1>
            <h2 style="color:blue">USER-NAME :<input type =
"text" name = "name"style="font-
size:12pt;color:blue"/></h2><br/><br/><br/>
            <h2 style="color:blue">PASSWORD  :<input
type="password" name = "pwd" style="font-
size:12pt;color:blue"/></h2><br/><br/><br/>
```

```
            <h2style="color:blue"><input type = "submit"
name = "submit" value = "submit" style="font-
size:12pt;color:blue" /></h2>
        </form>
        </body>
</html>
```

Loginpro.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%--
The taglib directive below imports the JSTL library. If you
uncomment it,
you must also add the JSTL library to the project. The Add
Library... action
on Libraries node in Projects view can be used to add the
JSTL 1.1 library.
--%>
<%--
<%@taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>

    <h1>LOGIN PAGE</h1>

   <%
        String name = request.getParameter("name");
        String password = request.getParameter("pwd");
        if(name.equals("puvi") && password.equals("ammu"))
        {
                session.setAttribute("username",name);
                response.sendRedirect("welcome.jsp");
        }
```

```
        else
        {
                response.sendRedirect("logo.html");
        }

        %>



    </body>
</html>
```

Welcome.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
        <title>WELCOME PAGE</title>
    </head>
    <body>
    <center style="color:orange"><h1>Welcome
    <%= session.getAttribute("username") %>
    </h1></center>
    </body>
</html>
```