**PRACTICAL # 11**

**OBJECT:**

**Java Server Pages (JSP)**

**THEORY:**

Java Server Pages (JSP) is Java based server-side programming platform that enables the creation of dynamic, platform-independent method for building Web-based applications. JSP can access all Java APIs, including the JDBC API to access databases.

Insert Java code in HTML pages by using JSP tags, most of which start with <% and end with %>.

Write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

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, to create dynamic web pages.

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

*→ JSP are always compiled before they are 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.*

**Setting up Development Environment**

1. To use JSP, you first need to setup Java JDK. Set appropriate classpath environment variables.

2. Next setup a web-server. A number of web servers support JSP, but we will use Tomcat server. After installing the web-server, setup appropriate classpaths.

**JSP Processing**

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. JSP page which ends with .jsp.

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.

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. The output is passed to the web server by the servlet engine.

The web server forwards the HTTP response to your browser in as HTML content.

JSP engine checks 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.

**JSP Lifecycle:**

Four steps in JSP Lifecycle:

**1. Compilation**

Parsing the JSP.

Turning the JSP into a servlet.

Compiling the servlet.

**2. 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:

public void jspInit(){

// Initialization code...

}

**3. Execution**

This phase of the JSP life cycle represents all interactions with requests until the JSP is destroyed.

void \_jspService(HttpServletRequest request, HttpServletResponse response) {

// Service handling code...

}

The \_jspService() method of a JSP is invoked on request basis. This 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, i.e, GET, POST, DELETE, etc.

**4. 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.

public void jspDestroy() {

// Your cleanup code goes here.

}

**JSP Elements**

**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.

**Syntax:**

*<% code fragment %>*

You can write the XML equivalent of the above syntax as follows −

*<jsp:scriptlet>*

*code fragment*

*</jsp:scriptlet>*

Example Page:

<html>

<head><title>Hello World</title></head>

<body>

Hello World!<br/>

<%

out.println("Your IP address is " + request.getRemoteAddr());

%>

</body>

</html>

**JSP Expression:**

<html>

<head><title>A Comment Test</title></head>

<body>

<p>Today's date: <%= (new java.util.Date()).toLocaleString()%></p>

</body>

</html>

**JSP Comments:**

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

<html>

<head><title>A Comment Test</title></head>

<body>

<h2>A Test of Comments</h2>

<%-- This comment will not be visible in the page source --%>

</body>

</html>

**ACTIVITIES**

**Activity 1**

Draw

**Activity 2**

Draw

**Activity 3**

Draw

**REVIEW QUESTIONS**

1. What
2. Which
3. Which
4. What
5. Which