UNIT III
jSP & ASP

Contents

- *Introduction*
- *Why Use JSP?*
- *Advantages of JSP*
- *Installing JSP*
- *JSP – Architecture*
- *JSP Processing*
- *JSP - Life Cycle*

What is Java Server Pages?

- Java Server 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 %>.
- 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.

Why Use JSP?

- 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

- **vs. Active Server Pages (ASP):** 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:** It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.

- **vs. Server-Side Includes (SSI):** 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 but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

- **vs. Static HTML:** Regular HTML, of course, cannot contain dynamic

Installing JSP

- First of all download JavaServer Web Development Kit (JSWDK1.0.1) from http://java.sun.com/products/servlet/download.html. JSWDK comes with full documentation and it's very easy to install.
- The JSWDK is the official reference implementation of the servlet 2.1 and JSP 1.0 specifications. It is used as a small stand-alone server for testing servlets and JSP pages before they are deployed to a full Web server that supports these technologies. It is free and reliable, but takes quite a bit of effort to install and configure.

Other Servers that support JSP

- Apache Tomcat.
  Tomcat is the official reference implementation of the servlet 2.2 and JSP 1.1 specifications. It can be used as a small stand-alone server for testing servlets and JSP pages, or can be integrated into the Apache Web server.

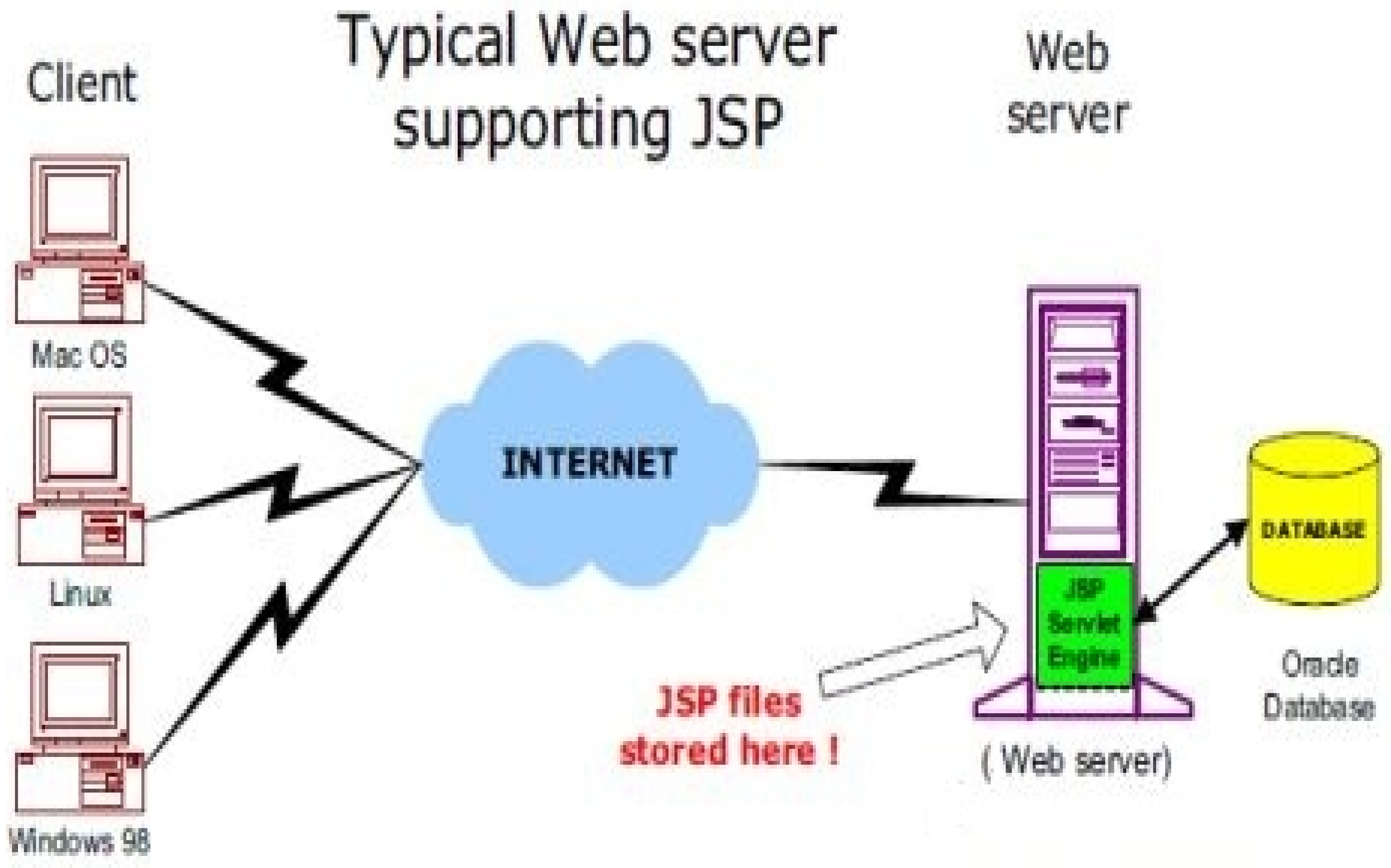
- Allaire JRun.
  JRun is a servlet and JSP engine that can be plugged into Netscape Enterprise or FastTrack servers, IIS, Microsoft Personal Web Server, older versions of Apache, O?Reilly?s WebSite, or StarNine WebSTAR.

- New Atlanta?s ServletExec.
  ServletExec is a fast servlet and JSP engine that can be plugged into most popular Web servers for Solaris, Windows, MacOS, HP-UX and Linux.

- Gefion's LiteWebServer (LWS). LWS is a small free Web server that supports servlets version **2.2** and JSP **1.1**.

- WebSphere. IBM's WebSphere very large application server now implements JSP.

JSP - Architecture



JSP Processing

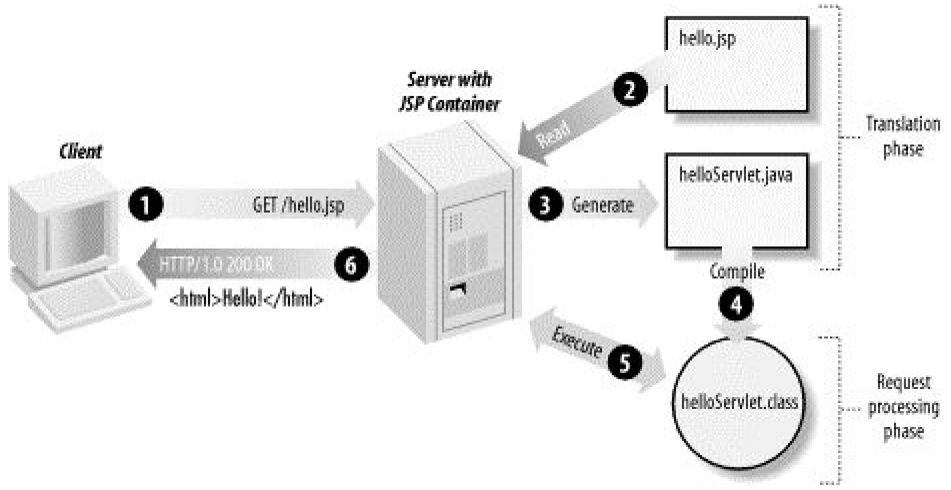
The following steps explain how the web server creates the web page using JSP:

- 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 behaviour of the page.

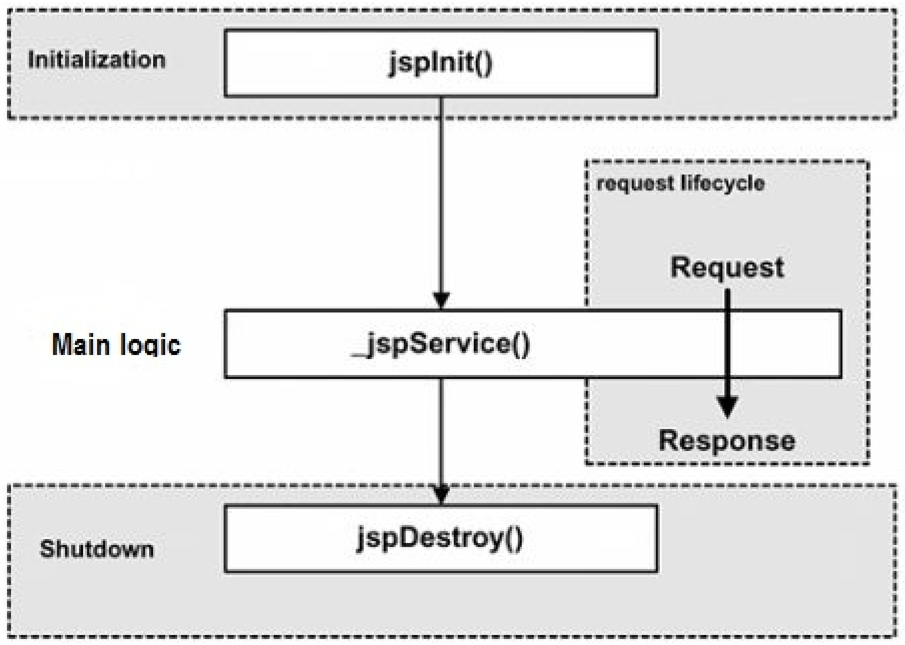
JSP Processing (contd..)

- 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.

Steps explained with the help of a diagram



JSP - Life Cycle



Paths followed by a JSP in its life cycle

The following are the paths followed by a JSP

1. Compilation
2. Initialization
3. Execution
4. Cleanup

- **(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:

- Parsing the JSP.
- Turning the JSP into a servlet.
- Compiling the servlet.

- **(2) JSP Initialization:**

When a container loads a JSP it invokes the jspInit() method before servicing any requests. 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 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.

JSP - Syntax

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:
- <% code fragment %>

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

<jsp:scriptlet>   code fragment</jsp:scriptlet>

Any text, HTML tags, or JSP elements you write must be outside the scriptlet.

Following is the syntax of Scriptlet:

```
<%
code fragment
 %>
```

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

```
<jsp:scriptlet>   code fragment</jsp:scriptlet>
```

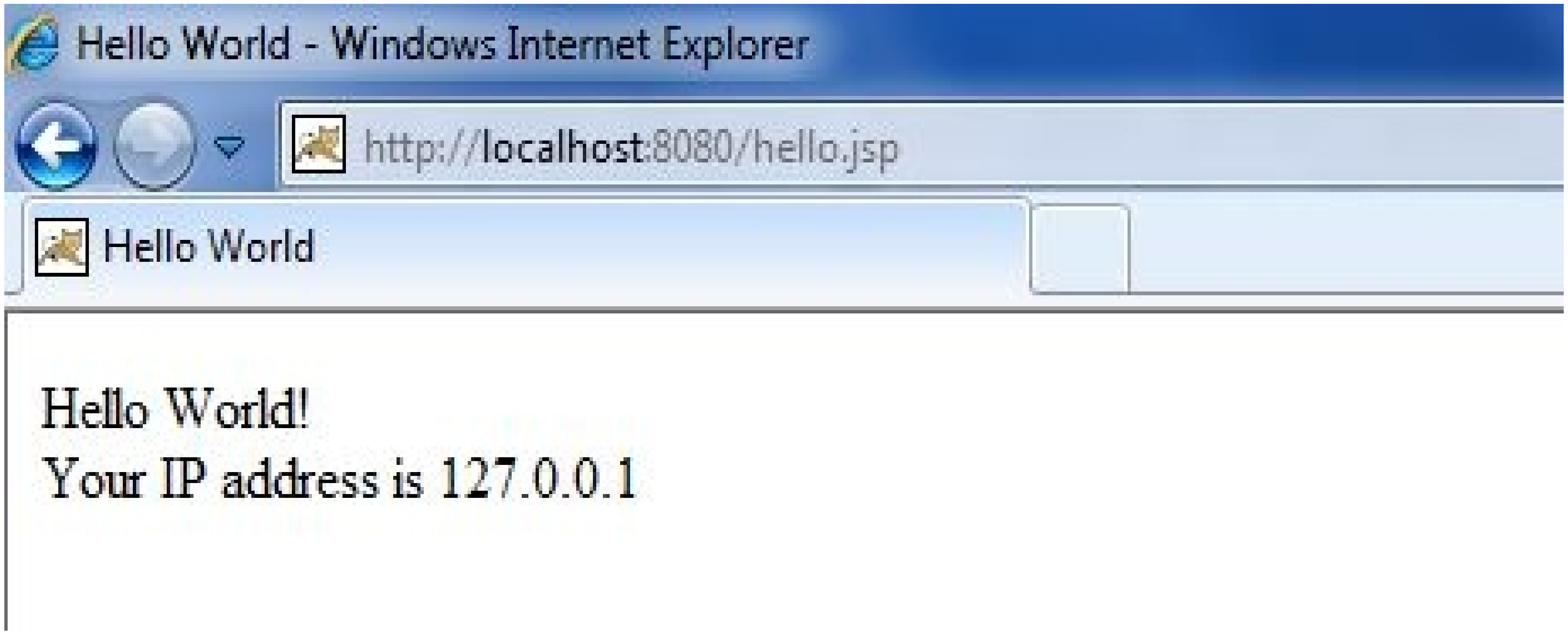
Any text, HTML tags, or JSP elements you write must be outside the scriptlet.

simple and first example for JSP

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
Hello World!<br/>
<%
out.println("Your IP address is " +
   request.getRemoteAddr());
%>
</body>
</html>
```

Output

Let us keep above code in JSP file hello.jsp and put this file in **C:\apache-tomcat-7.0.2\webapps\ROOT** directory and try to browse it by giving URL http://localhost:8080/hello.jsp. This would generate following result:


Hello World - Windows Internet Explorer
http://localhost:8080/hello.jsp
Hello World

```
Hello World!
Your IP address is 127.0.0.1
```

Another example

```
<html>
   <head>
   <title>First JSP page.</title>
   </head>
   <body>
   <p align="center"><font color="#FF0000" size="6">
<%="Java Developers Paradise"%>
</font></p>
   <p align="center"><font color="green" size="6">
<%="Hello JSP"%>
 </font></p>
   </body>
   </html>
```

Output

```
Java Developers Paradise
Hello JSP
```

JSP - Auto Refresh : Another Example

```
<%@ page import="java.io.*,java.util.*" %>
<html>
<head><title>Auto Refresh Header Example</title></head>
<body><center><h2>Auto Refresh Header Example</h2>
<%   // Set refresh, autoload time as 5 seconds
    response.setIntHeader("Refresh", 5);
  // Get current time
  Calendar calendar = new Calendar();
     int hour = calendar.get(Calendar.HOUR);
int minute = calendar.get(Calendar.MINUTE);
  int second = calendar.get(Calendar.SECOND);
    if(calendar.get(Calendar.AM_PM) == 0)
String CT = hour+":"+ minute +":"+ second ;
out.println("Crrent Time: " + CT + "\n");%>
</center></body></html>
```

Output

Auto Refresh Header Example

Current Time is: 9:44:50 PM

Application

- JSP enables a clean separation of business logic from presentation.
- JSP, by using Java as the scripting language, is not limited to a specific vendor platform.
- JSP, as an integral part of the J2EE architecture, has full access to server-side resources.

Scope of research

- Can JSP accept HTTP POST requests with multipart

form-data encode for file upload