JSP I

Now as we are aware about using servlets, let us understand the supportive technology in the form of JSP. Learning JSP is very easy as all the concepts are already known for servlets.

JSP as Complementary Technology

- JSP stands for Java Server Pages. JSP is built on top of servlets, but JSP is much easier to program. From programmer's point of view, JSP pages are simply HTML pages with embedded Java in them.
- Initially only servlet technology was available for developing dynamic web applications but later on JSP was developed as counter technology for servlets.
- JSP (JavaServer Pages) and servlets are both crucial technologies for building dynamic web applications in Java, but they serve different purposes and can be effectively used together as complementary technologies.

Servlets used for :

Server-side scripting: Handle core logic, manipulate data, and generate dynamic content.

Direct control: Provide fine-grained control over every aspect of the request-response cycle.

More complex: Require writing Java code, which can be less accessible for beginners.

JSP used for:

HTML with embedded Java: Combines static HTML markup with Java code snippets (scriptlets) for data and logic.

Easier development: Simplifies building UI and integrating dynamic content within familiar HTML syntax.

Less control: Offers less control compared to pure servlets, potentially limiting complex functionalities.

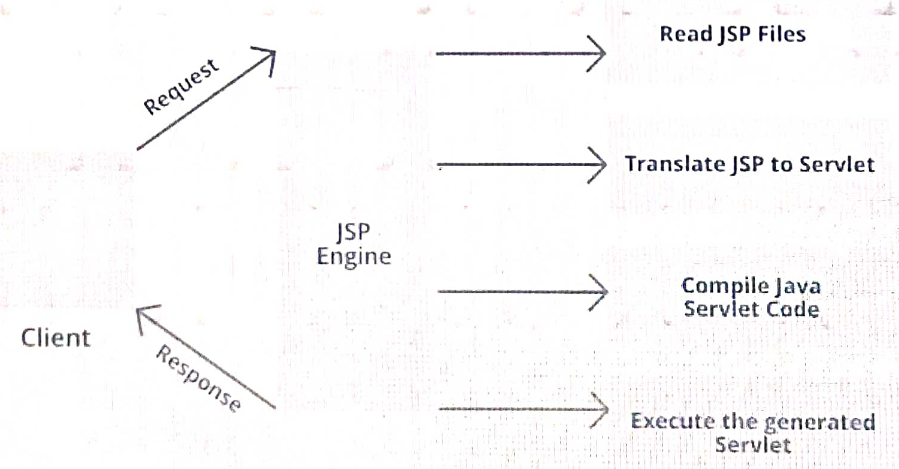
Difference between servlets and JSP

Servlets	JSP
Servlet is a pure Java code.	JSP is a tag based approach.
We write HTML in servlet code.	We write JSP code in HTML.
We cannot run Java script at client	We can run Java script at client side.

Difference between servlets and JSP

Servlets	JSP
Servlet is a pure Java code.	JSP is a tag based approach.
We write HTML in servlet code.	We write JSP code in HTML.
We cannot run Java script at client side.	We can run Java script at client side.
Servlets are tougher to code.	JSP coding is easier.
By default session management is not enabled. We are required to enable it.	In JSP session management is enabled by default..
Servlets do not have implicit objects.	JSP does have implicit objects viz. request, response, session, out etc.

JSP Life Cycle



The JSP life cycle refers to the process a JSP page goes through from its creation to its destruction. It involves several key stages

The JSP life cycle refers to the process a JSP page goes through from its creation to its destruction. It involves several key stages that ensure the proper execution and cleanup of the page.

1. Translation:

When a web server receives a request for a JSP page, it first checks if the page has already been translated into a servlet. If not, the JSP translator engine parses the JSP page and converts it into a Java servlet class.

This generated servlet class contains the equivalent Java code for the JSP tags and scriptlets in the original page.

2. Compilation:

The generated servlet class is then compiled into a Java bytecode class file. This bytecode is what the web server actually executes to serve the dynamic content of the JSP page.

3. Class Loading:

The compiled bytecode class is loaded into the web server's memory. This makes the generated servlet class available for execution.

4. Instantiation:

An instance of the generated servlet class is created. This instance will handle the specific request for the JSP page.

5. Initialization:

The jspInit() method of the generated servlet is called. This method allows the JSP page to perform any initialization tasks, such as connecting to databases or setting up beans.

6. Request Processing:

The _jspService() method of the generated servlet is called. This method is responsible for handling the actual request and generating the response. It executes the JSP code within the page, including interpreting scriptlets and evaluating expressions.

7. Destruction:

When the request is processed and the response is sent, the jspDestroy() method of the generated servlet is called. This method allows the JSP page to perform any cleanup tasks, such as closing database connections or releasing resources.

8. Unloading:

After a period of inactivity, the web server may unload the generated servlet class from its memory to free up resources. If another request comes in for the same JSP page, the life cycle will start again from the translation stage.

JSP Elements

JSP elements are the building blocks of JSP pages, allowing for dynamic content generation and interaction with Java code.

During translation, the JSP engine converts these elements into equivalent Java code within the generated servlet.

All the tags in HTML are valid to be written in JSP. HTML comment can even be added in JSP. In fact we can say that JSP is

JSP Elements

JSP elements are the building blocks of JSP pages, allowing for dynamic content generation and interaction with Java code. During translation, the JSP engine converts these elements into equivalent Java code within the generated servlet.

All the tags in HTML are valid to be written in JSP. HTML comment can even be added in JSP. In fact we can say that JSP is basically HTML and additional JSP elements.

1. Scriptlets (<% %>):

Contain Java code snippets embedded within the JSP page. Translated directly into Java code within the servlet's _jspService() method.

Example: <% int count = 10; %>

2. Expressions (<%= %>):

Evaluate Java expressions and print their results to the output stream. Translated into out.print() statements within the servlet's _jspService() method.

Example: Welcome back, <%= user.getName() %>!

3. Directives (<%@ %>):

Provide instructions to the JSP engine about various aspects of the page.

Common directives:

- page: Sets page-level attributes like language, import statements, error page, etc.
- include: Includes content from another file.
- taglib: Declares custom tag libraries.

Translated into appropriate Java code at the beginning of the generated servlet class.

4. Declarations (<%! %>):

Declare variables or methods at the class level, accessible throughout the JSP page. Translated into member variables or methods of the generated servlet class.

Example: <%! int total = 0; %>

5. JSP Actions (<jsp:action>):

Perform specific tasks like forwarding requests, including resources, or setting variables. Translated into Java code that calls the appropriate methods of the JSP engine.

Example: <jsp:forward page="result.jsp" />

6. JSP Standard Tag Library (JSTL) Tags:

Predefined set of tags for common tasks like iteration, conditional logic, formatting, etc. Translated into corresponding Java code from the JSTL tag library.

6. JSP Standard Tag Library (JSTL) Tags:

Predefined set of tags for common tasks like iteration, conditional logic, formatting, etc. Translated into corresponding Java code from the JSTL tag library.

Example: <c:forEach var="item" items="${products}">

7. Custom Tags:

User-defined tags for reusable functionality. Translated according to their implementation in a tag library handler class.

8. Comments (<%-- --%>):

Ignored by the JSP engine, not included in the generated servlet.

Creating Simple JSP

```
<%@ page  import="java.util.*"  language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Insert title here</title>
</head>
<body>
   <h1> Welcome to JSP </h1>
   <br/>
   This is template text
   <br/>
   <!--  This is HTML comment -->
   <br/>
   <%-- This is JSP comment --%>
   <br/>
   <%-- Scriptlet --%>
   <%
     int n = 0;
     out.print("Value of n : "+(++n));
   %>
   <br/>
   <% out.print(Calendar.getInstance().getTime()); %>
   <%-- Expression %>
```

Creating Simple JSP

```
<%@ page  import="java.util.*"  language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Insert title here</title>
</head>
<body>
   <h1> Welcome to JSP </h1>
   <br/>
   This is template text
   <br/>
   <!--  This is HTML comment -->
   <br/>
   <%-- This is JSP comment --%>
   <br/>
   <%-- Scriptlet --%>
   <%
     int n = 0;
     out.print("Value of n : "+(++n));
   %>
   <br/>
   <%  out.print(Calendar.getInstance().getTime()); %>
   <%-- Expression %>
   <%= Calendar.getInstance().getTime()  %>
   <br/>
   <%-- Declaration -->
   <%! int iVar;  %>
   <%= "value of iVar : "+(++iVar)  %>
</body>
</html>
```

At the time of transalation :

Scriptlet : Just symbols are striped off and code is inserted in service method of generated servlet. If a variable is declared in scriptlet, it gets initialized for every request

At the time of transalation :

Scriptlet : Just symbols are striped off and code is inserted in service method of generated servlet. If a variable is declared in scriptlet, it gets initialized for every request.

Expression : Note that no semicolon at the end because it is not java statement. It gets translated into java statement during translation and gets inserted in service method of generated servlet.

Declaration : Code in this JSP element gets inserted as a class member of generated servlet. If a variable is declared in declaration, it gets initialized only once when servlet is instantiated.

Assignments

1. Create a JSP to display a welcome message.
2. Use all the JSP elements in JSP to understand their usage and check how they get transformed in generated servlet.