created by Sun to allow easy creation of server side HTML pages.

JSP

1) JSP stands for Java Server Pages.
2) It is a server side technology.
3) It is used for creating web application.
4) It is used for creating dynamic web content.
5) JSP tags are used to insert JAVA code into HTML pages
6) It is an advanced version of Servlet Technology
7) It helps to create dynamic and platform independent web pages
8) Java code can be inserted in HTML / XML pages or both.
9) JSP is first converted into servlet by JSP container before processing the client's request.
10) They are easy to maintain
11) JSP are extended version of Servlet

JSP Syntax

1) Declaration Tag : Used to declare page wide variables and methods or functions.

Syntax : `<%! Dec var %>`

`<%! int var = 10; %>`

JSP lies in the presentation tier on the web server with the main responsibility of generating HTML content that needs to be served to the browser. It also has the additional responsibility of pass on the requests to the backend through the JavaBeans as and when required

2) Java Scriplets

It allows us to add any number of Java code, variables & expressions into JSP (valid)

Syntax `<% java code %>`

3) JSP Expression (useful shorthand for printing out strings and contents of variables)

It evaluates and convert the expression to a string. (run-time)

Syntax `<%= expression %>`

`<% num1 = num1 + num2 %>`

`<% = new java.util.Date() %>`

4) JAVA Comments

It contains the text that is added for information which has to be ignored

Syntax `<% -- JSP comments %>`

Advantages of JSP

Inherit platform independence

High performance

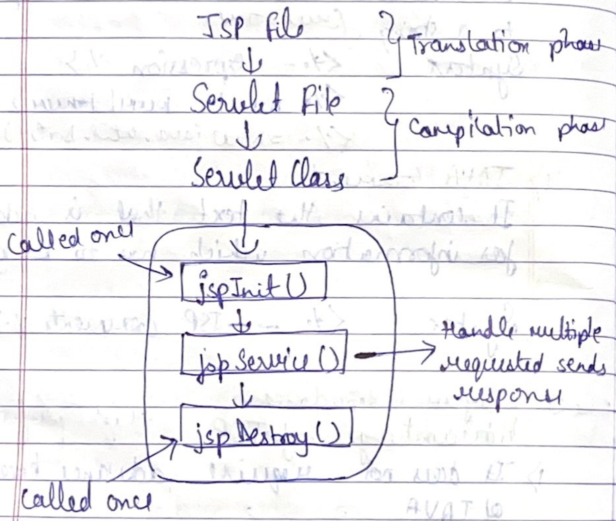
1) It does not require advance knowledge of JAVA
2) It is capable of handling exceptions
3) Easy to learn and use
4) Implicit objects are there which reduces the length of code
5) Separates presentation from content

Convenient to code and maintain against Servlet

directives → affect the overall structure of the servlet class generated from this JSP
<%@ directive attribute="value" %>
two types → page & include

Disadvantages
1) Difficult to debug for errors
2) It's output is HTML which lacks features

JSP Life Cycle



Translation of JSP page to servlet
This is the first step of JSP life cycle. Here test.jsp file is translated to test.java

Actions → control the behavior of the servlet engine, can dynamically insert a file,
jsp:include jsp:useBean
jsp:forward

Compilation of JSP page
Here the generated java servlet file test.java is compiled to class file, test.class

Classloading
Servlet class which gets loaded from JSP will be loaded into the container

Instantiation
Here the instance of class is generated.

Initialization
jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.

Request processing
jspService() method is used to serve the raised requests by JSP. It takes requests and response object as parameters. This method cannot be overridden.

JSP CleanUp
jspDestroy() method is called once to remove the JSP from the container. It can be overridden.

Date.
Page No.

Architecture – flow of JSP request

HTTP request comes to a web server

The server extension receives the request and passes it on to the JSP engine

The JSP engine invokes JSP parser to parse the JSP and check for any syntax errors

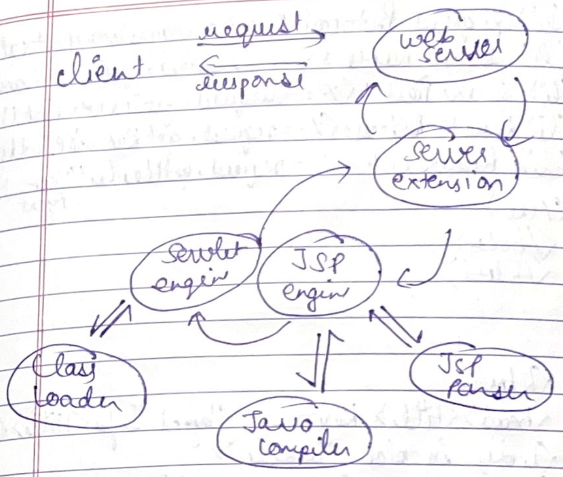
On successful parsing, the JSP engine invokes the Java compiler to compile the JSP into an equivalent servlet class

Once the class is generated, the control is passed on to a servlet engine

The servlet engine loads the servlet into memory using its class loader

The appropriate methods in the servlet are invoked and the response is routed back to the browser via the server extension and web server

Date.
Page No.



Scripting Elements

[A] Expressions
There are no of predefined variables
1) request
2) response
3) session
4) out

1)
```
<html>
<head><title> Request Header Info </title></head>
<body bgcolor="white">
<ul>
<li> Request Method : <%= request.getMethod()%>
<li> Request URL : <%= request.getRequestURI()%>
```

```
<li> Request Protocol : <%= request.getProtocol()%>
<li> server name : <%= request.getServerName() %>
<li> Server Port : <%= request.getServerPort() %>
<li> Remote Address : <%= request.getRemoteAddr() %>
<li> Browser : <%= request.getHeader("User-Agent") %>
</ul>
 </body>
 <html>
```

2)
```
 <html>
 <head><title> Response Object Info </title></head>
<body bgcolor = "white">
<% response.setContentType ("text/html"); %>

Buffer size : <response: getBufferSize() %> bytes
                                        <br/>

Character encoding : <%= response.getCharacterEncoding() %> <br/>

Locale : <%= response.getLocale() %>

<%-- redirect the user to another html --%>

<% response.sendRedirect ("http://localhost:8080/
                          sendthen.html"); %>
 </body>
 </html>
```

3)
```
<html
<head><title> Session Object Info </title></head>
<body bgcolor ="white">

Session : <%= session.getId() %> <br/>
<% session.invalidate(); %>

Session : <% if (session.getId() != null)
          out.println (session.getId());
          else
          out.println ("session ended"); %>
</body>
</html>
```

4)
```
<html>
<head><title> Out Object Info </title></head>
<body bgcolor= "white">
<% out.print ("<h3> JSP BufferInformation </h3>");

<%-- display the page buffer size --%>
Buffer: <%=out.getBufferSize() %> bytes <br/>

<%-- display the AutoFlash setting --%>
AutoFlash: <%=out.isAutoFlash() %> <br/>

<% --display the free page buffer space --%>
Remaining buffer: <%=out.getRemaining() %> bytes
</body>
</html>
```

Date.
Page No.

[B] Scriplets
<% and %> tags - code reside b/w this tag

Code that is defined within a scriplet can access any variable that have been declared

Are like declarations, always use ; to end statements & expression, can have multiple expression & statements.

~~B~~ code goes into the service method of the JSP's compiled servlet which mean it is executed only once when a request is actually serviced by the JSP.

```
<html> <~~body~~ head> <title>. </title> </head>
<body bg color="white">

<% java.util.Date now=new java.util.Date(); %>
Current da~~te~~ & amp time :<%=now %>

<% if (now.getHours () <12) { %> GM! <% }
else if (now.getHours()<17 {%> GA! <% }
else { %> GE! <% } %>
</body>
</html>
```

Date.
Page No.

[C] Declaration <%! Java Code %>
used to define methods & fields that get inserted into the main body of the servlet class

scope → JSP file, but if JSP file includes other files with the include directive the scope expands to cover the included files as well.

```
<html>
<body>
<%! int counter=0; %>

Global counter : <%= ++counter %>
</body>
</html>
```

Directive Elements <% @directive attribute="value" %>
Affect the overall structure of the servlet class generated from this JSP

Page Directives
define attributes that apply to an entire JSP page

Let you do things like

import classes
handle error message
define if the JSP is thread-safe
define is session object is available
Set the page content type

Include Directive
Inserts the contents of another file in the main JSP file, where the directive is located

Useful for including copyright info scripting language files, or anything you might want to reuse in other applications.

The include file can be an HTML file, a JSP file, a text file, or a code file written in the Java prog. Lang.

(D) Actions
Control the behavior of the servlet engine can dynamically insert a file

jsp:include
jsp:forward
jsp:useBean

jsp:include Action
Lets you insert files into the page being generated

<jsp:include page="relative URL" />

Unlike the include directive, which inserts the file at the time the JSP page is translated into a servlet, this action inserts the file at the time the page is requested.

jsp:forward Action
Forwards a client request to an HTML file, JSP file, or servlet for processing

<jsp:forward page="relativeURL"/>