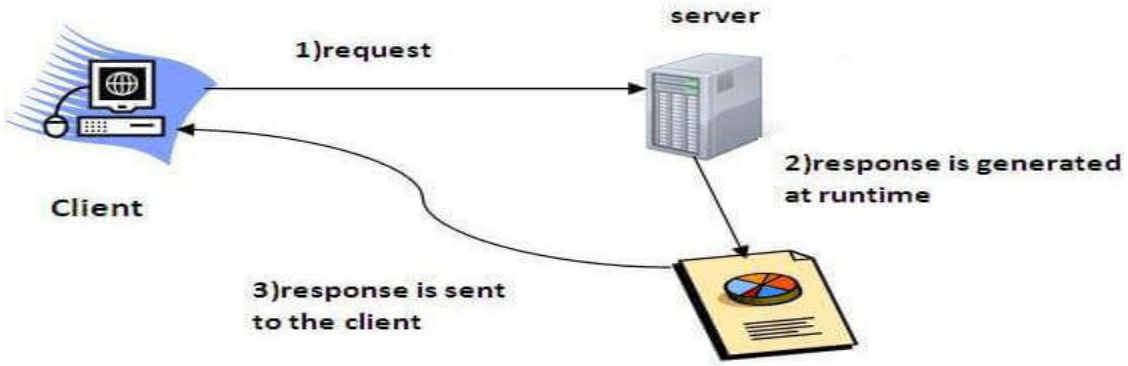
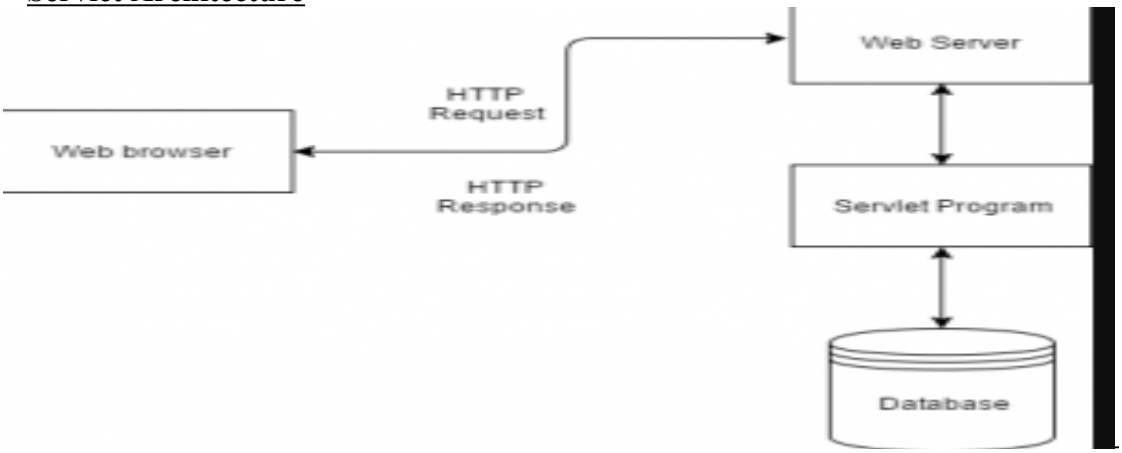
UNIT 5

SERVLET

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.
- There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.



Servlet Architecture



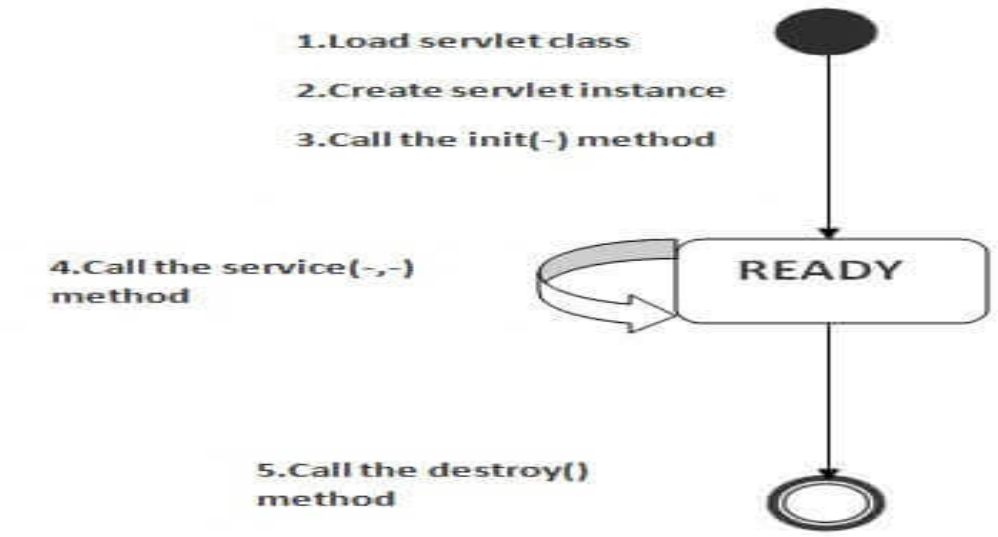
- The clients send the request to the web server.
- The web server receives the request.
- The web server passes the request to the corresponding servlet.
- The servlet processes the request and generate the response in the form of output.
- The servlet send the response back to the web server.
- The web server sends the response back to the client and the client browser displays it on the screen.

Life Cycle of a Servlet (Servlet Life Cycle)

UNIT 5

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

- Servlet class is loaded.
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.



As displayed in the above diagram,

- There are three states of a servlet: new, ready and end.
- The servlet is in new state if servlet instance is created.
- After invoking the init() method, Servlet comes in the ready state.
- In the ready state, servlet performs all the tasks.
- When the web container invokes the destroy() method, it shifts to the end state.

Servlet Interface

Servlet interface provides common behavior to all the servlets. Servlet interface defines methods that all servlets must implement. Servlet interface needs to be implemented for creating any servlet (either directly or indirectly). It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

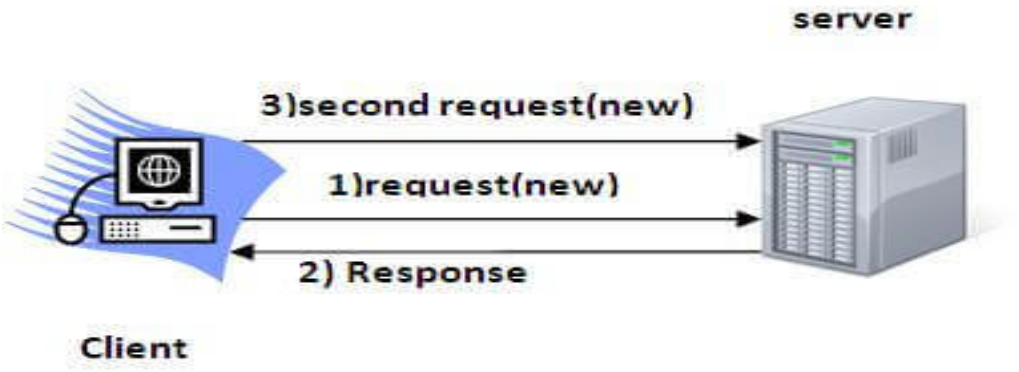
There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

UNIT 5

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

Session Tracking in Servlets

- **Session** simply means a particular interval of time.
- **Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.
- Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.
- HTTP is stateless that means each request is considered as the new request.



Why use Session Tracking?
To recognize the user It is used to recognize the particular user.

Session Tracking Techniques
There are four techniques used in Session tracking:

UNIT 5

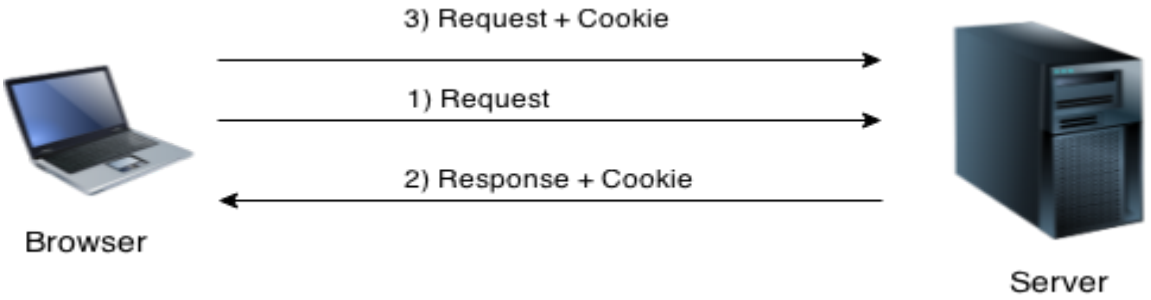
- **Cookies**
- **Hidden Form Field**
- **URL Rewriting**
- **HttpSession**

Cookies in Servlet

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Types of Cookie

There are 2 types of cookies in servlets.

- Non-persistent cookie
- Persistent cookie

Non-persistent cookie- **It is** valid for single session **only. It is removed each time when user closes the browser.**

Persistent cookie- **It is** valid for multiple session **. It is not removed each time when user closes the browser. It is removed only if user logout or signout.**

Advantage of Cookies

- Simplest technique of maintaining the state.
- Cookies are maintained at client side.

Disadvantage of Cookies

- It will not work if cookie is disabled from the browser.
- Only textual information can be set in Cookie object.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

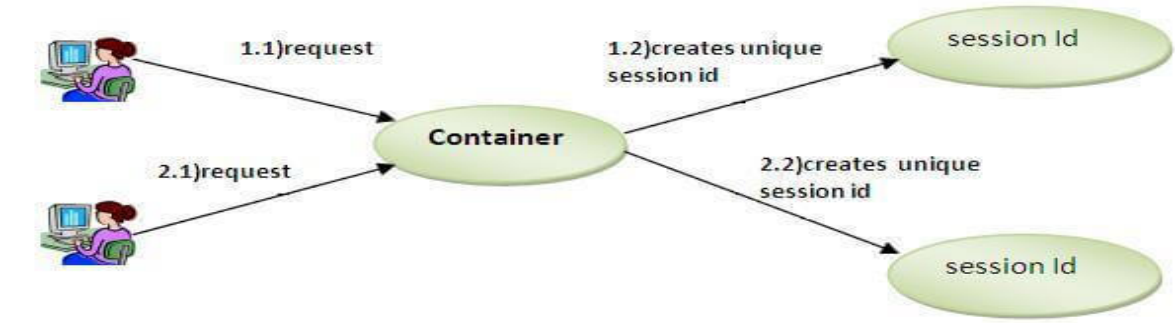
UNIT 5

Cookies	Sessions
Cookies are stored in browser as text file format.	Sessions are stored in server side.
It is stored limit amount of data.	It is stored unlimited amount of data
It is only allowing 4kb[4096bytes].	It is holding the multiple variable in sessions.
It is not holding the multiple variable in cookies.	It is holding the multiple variable in sessions.
we can accessing the cookies values in easily. So it is less secure.	we cannot accessing the session values in easily.So it is more secure.
setting the cookie time to expire the cookie.	using session_destory(), we we will destroyed the sessions.
The setcookie() function must appear BEFORE the <html> tag.	The session_start() function must be the very first thing in your document. Before any HTML tags.

HttpSession interface

In such case, container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks:

- bind objects
- view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



Servlet program for servlet login and logout using cookies

Login

<!DOCTYPE html>

UNIT 5

```
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,       initial-scale=1.0">
   </head>
   <body>
      <form action="LoginServlet"  method="post">
         name:<input type="text" name="name"><br><br>
         Password:<input type="password" name="password"><br>          <br>
         <input type="submit" name="go">
      </form>
       </body>
</html>
```

Logout

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author DELL
 */
@WebServlet(urlPatterns = {"/logoutServlet"})
public class logoutServlet extends HttpServlet {

   @Override
   protected void doGet(HttpServletRequest request,                    HttpServletResponse
response)
         throws ServletException, IOException {


      response.setContentType("text/html");
      PrintWriter out=response.getWriter();

      request.getRequestDispatcher("link.html").include(request, response);

      Cookie ck=new Cookie("name","");
      ck.setMaxAge(0);
      response.addCookie(ck);
```

```
        out.println("<br><br>you have Successfull logout");

        out.close();
    }

}
```

Handling GET and POST Requests
To handle HTTP requests in a servlet, extend the HttpServlet class and override the servlet methods that handle the HTTP requests that your servlet supports. **The methods that handle these requests are doGet and doPost.**

Handling GET requests- Handling GET requests involves overriding the doGet method.
Handling POST Requests- Handling POST requests involves overriding the doPost method.

- Like doGet () method, the do Post () method is invoked by server through service () method to handle HTTP POST request.
- The doPost () method is used when large amount of data is required to be passed to the server which is not possible with the help of doGet () method.
- In doGet () method, parameters are appended to the URL whereas, in do Post () method parameters are sent in separate line in the HTTP request body.
- The do Get ( ) method is mostly used when some information is to be retrieved from the server and the doPost () method is used when data is to be updated on server or data is to be submitted to the server.

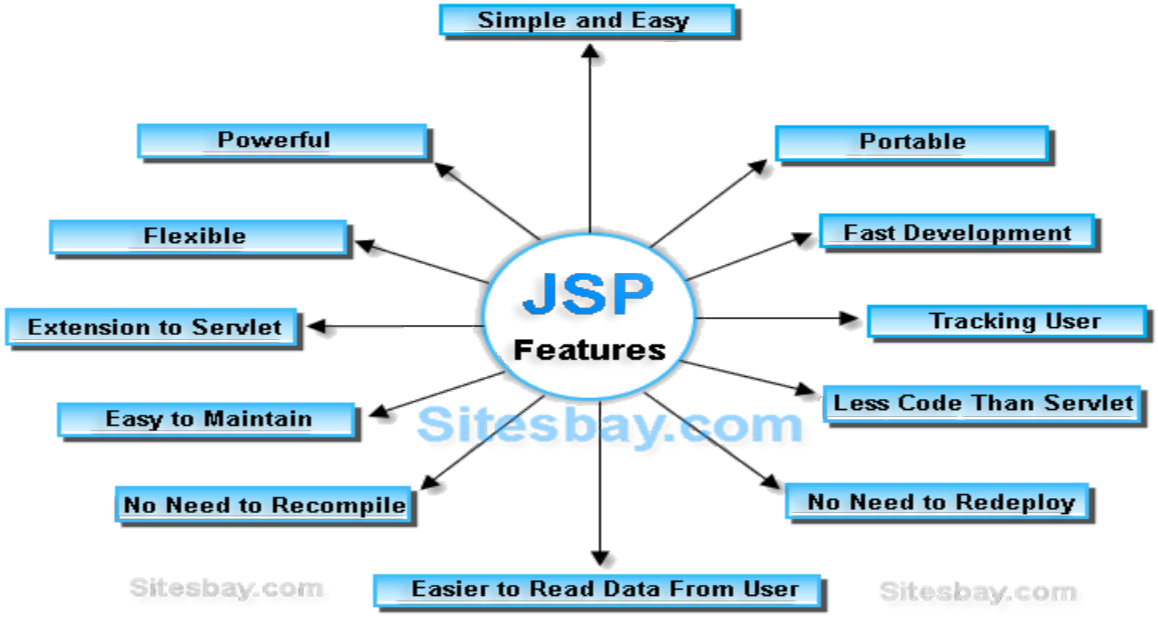
JSP

- **JSP** technology is used to create dynamic web applications.
- **JSP** pages are easier to maintain then a **Servlet**.
- JSP pages are opposite of Servlets as a servlet adds HTML code inside Java code, while JSP adds Java code inside HTML using JSP tags.
- Everything a Servlet can do, a JSP page can also do it.
- Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.
- JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.
- **Using JSP, one can easily separate Presentation and Business logic** as a web designer can design and update JSP pages creating the presentation layer and java developer can write server side complex computational code without concerning the web design. And both the layers can easily interact over HTTP requests.

Why JSP is preffered over servlets?
- JSP provides an easier way to code dynamic web pages.
- JSP does not require additional files like, java class files, web.xml etc

UNIT 5

- Any change in the JSP code is handled by Web Container(Application server like tomcat), and doesn't require re-compilation.
- JSP pages can be directly accessed, and web.xml mapping is not required like in servlets.



Main features of JSP

- **Make interactive websites**.
  One of the main uses of JSP is to make interactive webpages. Webpages that not just static but alive, **dynamic and are able to interact with the user in real time**.

- **Easier to read data from user**.
  User interacting with JSP controls like *textbox, button, dropdown list, checkbox* can enter some information. **Using JSP, it is easier to read this information entered by the user and send it to the server.** For example, A user entering a form and clicking the "submit" button. Using JSP, the data entered in a form is read(*when the submit button is clicked*) and sent to the server side for further processing.
    - **Easier to display server response**.

  After the data read from client's input is sent to the server, its response is sent back to the client and it is easily displayed. For example, after a user submits a form, a response from the server is displayed using JSP, such as - "Thanks for filling the form".

- **Allows to add Java to your website.**
  As you know JSP stands for *Java Server Pages*, one of the main feature of **JSP is to allow Java code to be added in between your HTML code, in order to give our webpage the power of Java and make it further powerful and interactive**. JSP page is internally converted to a *bytecode* java file, hence all the Java features like security, flexibility, platform independent are available to JSP as well.

- **Easier to connect to the database.**

UNIT 5

One of the main features of JSP is to easily **allow us to connect our website with the database, so that we can send the data entered by each user to a database and read it back from database, when required.**

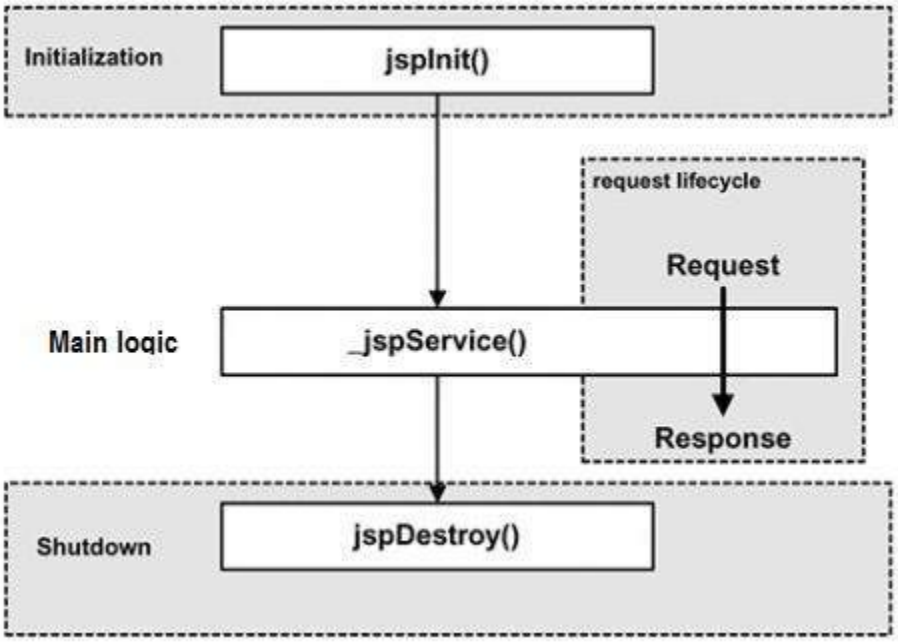
- **Tracking the User.**
  JSP allows us to track the selections made by the user during user interaction with the website by maintaining the information in the ***Session*** object, ***Application*** object or in the ***Cookies***(*storing user specific information on the user's system*. This can be helpful when you need to track one or multiple users or the selections made by a user during the current interaction with the website.
- **Easy to code.**

JSP - Lifecycle

A JSP life cycle is defined as the process from its creation till the destruction.
The following are the paths followed by a JSP −

- **Compilation**
- **Initialization**
- **Execution**
- **Cleanup**

The four major phases of a JSP life cycle are very similar to the Servlet Life Cycle. The four phases have been described below −



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.

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 −

```
public void jspInit(){
  // Initialization code...
}
```

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 as follows −

```
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.

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 −

```
public void jspDestroy() {
  // Your cleanup code goes here.
}
```

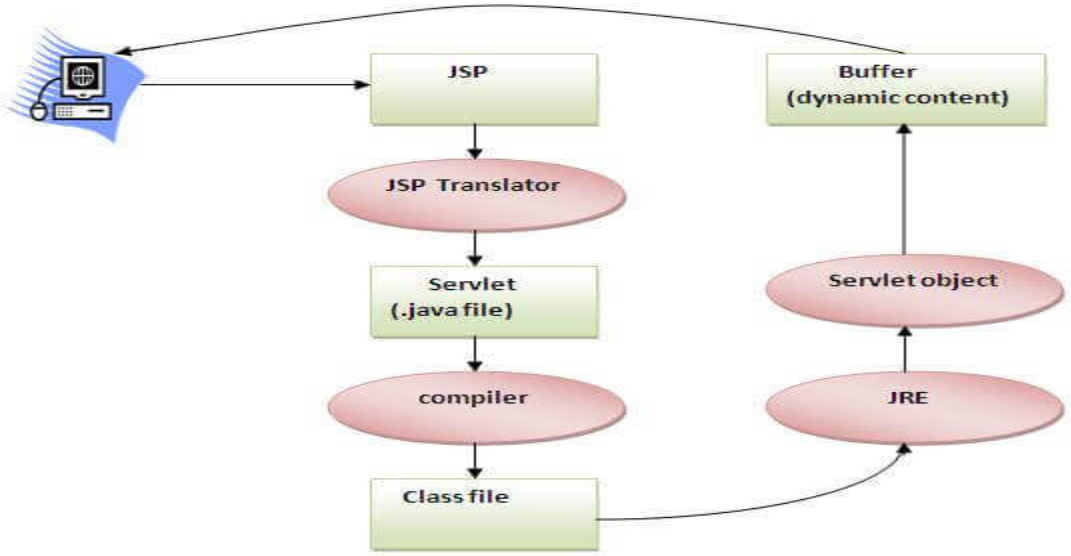
The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (the container invokes jspInit() method).
- Request processing (the container invokes _jspService() method).
- Destroy (the container invokes jspDestroy() method).

Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.

UNIT 5



As depicted in the above diagram,

- JSP page is translated into Servlet by the help of JSP translator.
- The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet.
- After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

Creating a simple JSP Page

To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page.

index.jsp

Let's see the simple example of JSP where we are using the scriptlet tag to put Java code in the JSP page.

```
<html>
<body>
<% out.print(2*5); %>
</body>
</html>
```

It will print **10** on the browser.

Follow the following steps to execute this JSP page:

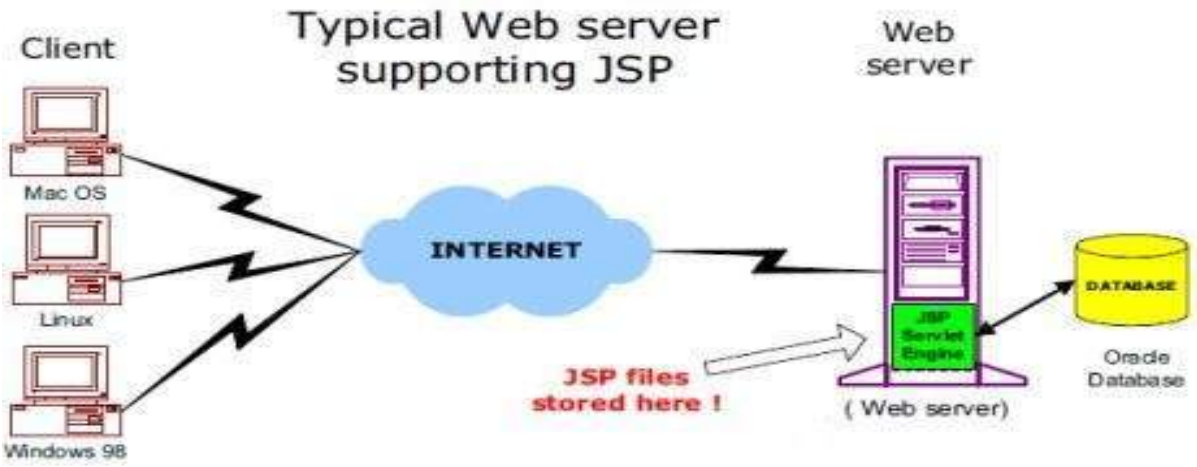
- Start the server
- Put the JSP file in a folder and deploy on the server
- Visit the browser by the URL http://localhost:portno/contextRoot/jspfile, for example, http://localhost:8888/myapplication/index.jsp

UNIT 5

JSP PROCESSING- JSP - Architecture

The web server needs a JSP engine, i.e, a container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. This tutorial makes use of Apache which has built-in JSP container to support JSP pages development.

A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

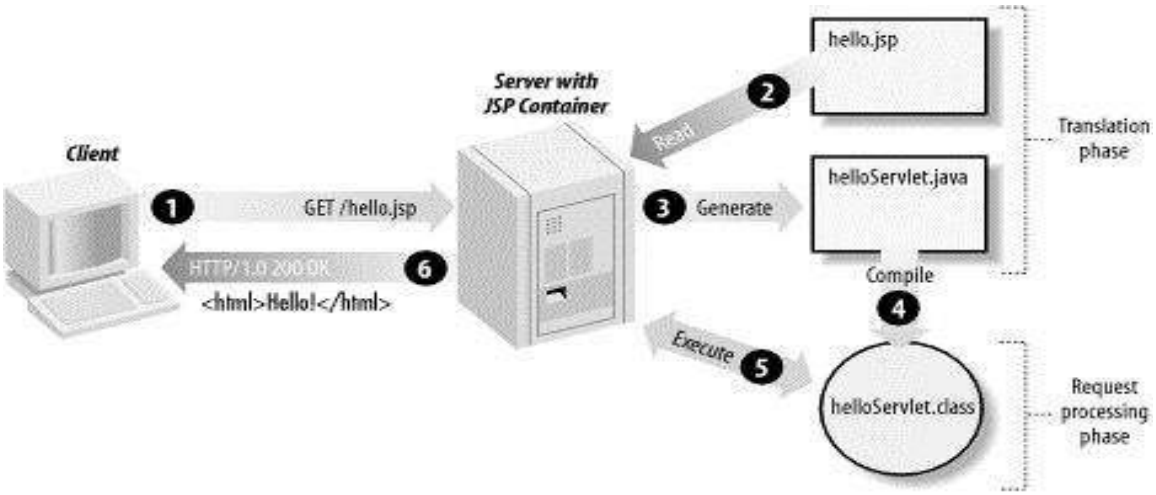


JSP Processing

The following steps explain how the web server creates the Webpage 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. This code 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. The output is furthur passed on to the web server by the servlet engine inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

UNIT 5



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 the 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.

JSP SYANTAX
JSP DIRECTIVES
DECLARATIONS
SCRIPLETS
EXPRESSIONS
STANDARD ACTION

JSP DIRECTIVES

These directives provide directions and instructions to the container, telling it how to handle certain aspects of the JSP processing.

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

<%@ directive attribute = "value" %>

Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.

The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

There are three types of directive tag −

S.No.	Directive & Description
1	**<%@ page ... %>** Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.

2	**<%@ include ... %>** Includes a file during the translation phase.
3	**<%@ taglib ... %>** Declares a tag library, containing custom actions, used in the page

JSP ACTION

These actions use constructs in XML syntax **to control the behavior of the servlet engine.** You can dynamically insert a file, reuse JavaBeans components, forward the user to another page, or generate HTML for the Java plugin.

There is only one syntax for the Action element, as it conforms to the XML standard −

<jsp:action_name attribute = "value" />

Action elements are basically predefined functions. The following table lists out the available JSP actions −

S.No.	Syntax & Purpose
1	**jsp:include** **Includes a file at the time the page is requested.**
2	**jsp:useBean** **Finds or instantiates a JavaBean.**
3	**jsp:setProperty** **Sets the property of a JavaBean.**
4	**jsp:getProperty** **Inserts the property of a JavaBean into the output.**
5	jsp:forward Forwards the requester to a new page.
6	jsp:plugin Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin.
7	jsp:element Defines XML elements dynamically.
8	**jsp:attribute** **Defines dynamically-defined XML element's attribute.**
9	**jsp:body** **Defines dynamically-defined XML element's body.**
10	**jsp:text**

UNIT 5

	Used to write template text in JSP pages and documents.

IMPLICIT OBJECT IN JSP

These Objects are the Java objects that the JSP Container makes available to the developers in each page and the developer can call them directly without being explicitly declared. JSP Implicit Objects are also called pre-defined variables.

Following table lists out the nine Implicit Objects that JSP supports −

S.No.	Object & Description
1	**Request** **This is the HttpServletRequest object associated with the request.**
2	**Response** **This is the HttpServletResponse object associated with the response to the client.**
3	**Out** **This is the PrintWriter object used to send output to the client.**
4	**Session** **This is the HttpSession object associated with the request.**
5	**Application** This is the **ServletContext** object associated with the application context.
6	**Config** **This is the ServletConfig object associated with the page.**
7	**pageContext** **This encapsulates use of server-specific features like higher performance JspWriters.**
8	**Page** **This is simply a synonym for this, and is used to call the methods defined by the translated servlet class.**
9	**Exception** The **Exception** object allows the exception data to be accessed by designated JSP.

JSP Scriptlet tag (Scripting elements)

In JSP, java code can be written inside the jsp page using the scriptlet tag.

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag

UNIT 5

- expression tag
- declaration tag

JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

<% java source code %>

JSP expression tag

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

<%= statement %>

JSP Declaration Tag

- The **JSP declaration tag** is used *to declare fields and methods*.
- The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.
- So it doesn't get memory at each request.

Syntax of JSP declaration tag

<%! field or method declaration %>

Custom Tags in JSP

- **Custom tags** are user-defined tags.
- They eliminates the possibility of scriptlet tag and separates the business logic from the JSP page.
- The same business logic can be used many times by the use of custom tag.

Advantages of Custom Tags

- **Eliminates the need of scriptlet tag** The custom tags eliminates the need of scriptlet tag which is considered bad programming approach in JSP.
- **Separation of business logic from JSP** The custom tags separate the the business logic from the JSP page so that it may be easy to maintain.
- **Re-usability** The custom tags makes the possibility to reuse the same business logic again and again.

Syntax to use custom tag

There are two ways to use the custom tag. They are given below:

<prefix:tagname attr1=value1....attrn=valuen **/>**

OR

<prefix:tagname attr1=value1....attrn=valuen **>**
body code

UNIT 5

</prefix:tagname>

Difference between Servlet & JSP

Servlet	JSP
Servlet is a java code.	JSP is a html based code.
Writing code for servlet is harder than JSP as it is html in java.	JSP is easy to code as it is java in html.
Servlet plays a controller role in MVC approach.	JSP is the view in MVC approach for showing output.
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in JSP lifecycle is the translation of JSP to java code and then compile.
Servlet can accept all protocol requests.	JSP only accept http requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
In Servlet by default session management is not enabled, user have to enable it explicitly.	In JSP session management is automatically enabled.
In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.	In JSP business logic is separated from presentation logic by using javaBeans.
Modification in Servlet is a time consuming task because it includes reloading, recompiling and restarting the server.	JSP modification is fast, just need to click the refresh button.

Assignment

Q1. Explain lifecycle of Servlet & JSP.
Q2. Define JSP. Explain different types of JSP Directives.
Q3. Explain JSP Action tags in detail(atleast six).

UNIT 5

Q4. Write the difference between the following:

- JSP & Servlet
- Session & Cookies

Q5. Explain JSP Processing & implicit objects in it.

Q6. Explain cookie and http-session with respect to session tracking.