Module 3

Q What is server?

1. A servlet is a Java programming class used to create dynamic web content.
2. Servlets work on the server side.
3. Servlet runs on a web server (or servlet container) to handle requests and generate responses.
4. They are used to handle the request obtained from the web server, process the request, produce the response, and then send a response back to the web server.
5. Servlets are capable of handling complex requests obtained from the web server.

Types of Servlet

- **Generic Servlets**: These are those servlets that provide functionality for implementing a servlet. It is a generic class from which all the customizable servlets are derived. It is protocol-independent and provides support for HTTP, FTP, and SMTP protocols. The class used is '**javax.servlet.Servlet'** and it only has 2 methods – init() to initialize & allocate memory to the servlet and destroy() to deallocate the servlet.
- **HTTP Servlets**: These are protocol dependent servlets, that provides support for HTTP request and response. It is typically used to create web apps. And has two of the most used methods – doGET() and doPOST() each serving their own purpose.

Q. Explain Java servlet Architecture.

Servlet Architecture contains the business logic to process all the requests made by client.

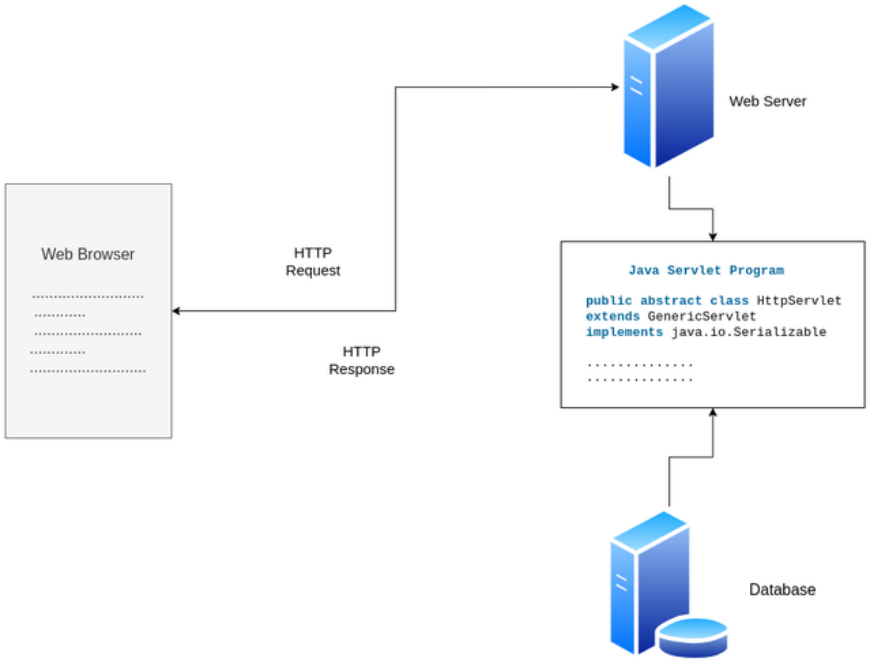


Fig : Java Servlet Architecture

Components of Servlet Architecture

1. Client

The client shown in the architecture above is the web browser and it primarily works as a medium that sends out HTTP requests over to the web server and the web server generates a response based on some processing in the servlet and the client further processes the response.

2. Web Server

Primary job of a web server is to process the requests and responses that a user sends over time and maintain how a web user would be able to access the files that has been hosted over the server. The server we are talking about here is a software which manages access to a centralized resource or service in a network. There are precisely two types of webservers:

- Static web server
- Dynamic web server

3. Web Container

Web container is another typical component in servlet architecture which is responsible for communicating with the servlets. Two prime tasks of a web container are:

- Managing the servlet lifecycle
- URL mapping

Execution of Java Servlets

Execution of Servlets basically involves Six basic steps:

1. The Clients send the request to the Web Server.
2. The Web Server receives the request.
3. The Web Server passes the request to the corresponding servlet.
4. The Servlet processes the request and generates the response in the form of output.
5. The Servlet sends the response back to the webserver.
6. The Web Server sends the response back to the client and the client browser displays it on the screen.

Q. Explain servlet life cycle.

The web container maintains the life cycle of a servlet instance.

1. Servlet class is loaded.
2. Servlet instance is created.
3. Init() method is invoked.
4. Service() method is invoked.
5. Destroy() method is invoked.

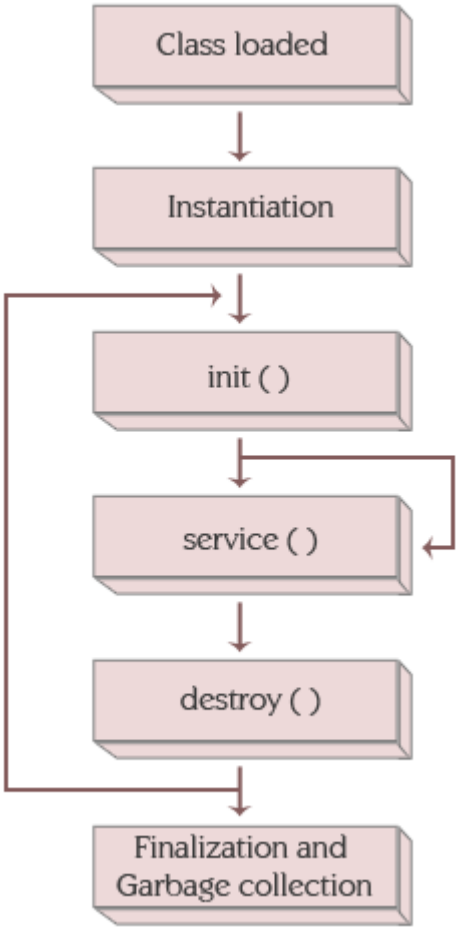


Fig : Servlet Life Cycle

1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init() method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet.

4) service() method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. This method

determines the type of HTTP request (GET, POST, etc.) and calls the appropriate method (doGet(), doPost(), etc.)

doGet() : The doGet() method in servlets is used to process the HTTP GET requests. So, basically, the HTTP GET method should be used to get the data from the server to the browser.

doPost() : The doPost() method in servlets is used to process the HTTP POST requests. It is used to submit the data from the browser to the server for processing.

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc.

Q. Difference between HTTP GET and POST method.

GET	POST
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3) Get request can be bookmarked.	Post request cannot be bookmarked.
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent.
5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.
6) Request made through GET method are stored in Browser history.	Request made through POST method is not stored in Browser history.
7) Request made through GET method are stored in cache memory of Browser.	Request made through POST method are not stored in cache memory of Browser.
8) In GET method only ASCII characters are allowed.	In POST method all types of data is allowed.
9) GET requests are only used to request data (not modify)	POST requests can be used to create and modify data.

Q. Explain session Handling and its Techniques.

1. **Session** simply means a particular interval of time.
2. **Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.
3. 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.

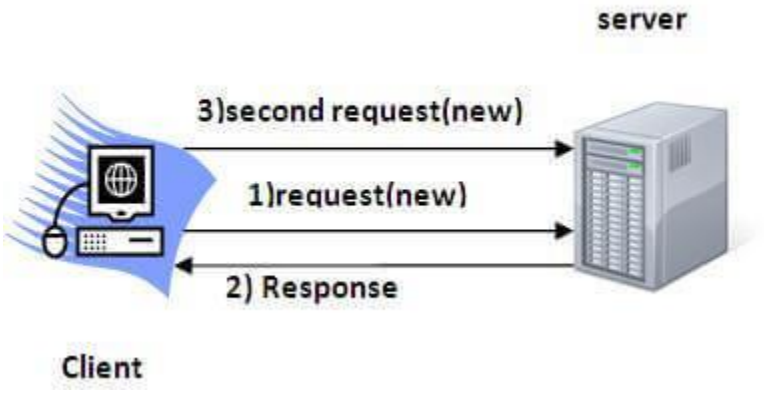


Fig : HTTP request and response

Session Tracking Techniques

There are four techniques used in Session tracking:

1. Cookies
2. Hidden Form Field
3. URL Rewriting
4. HttpSession

1. Cookies

- Cookies are the textual information that is stored in key-value pair format to the client's browser during multiple requests.
- It is one of the state management techniques in session tracking.
- Basically, the server treats every client request as a new one so to avoid this situation cookies are used.
- When the client generates a request, the server gives the response with cookies having an id which are then stored in the client's browser. Thus if the client generates a second request, a cookie with the matched id is also sent to the server. The server will fetch the cookie id, if found it will treat it as an old request otherwise the request is considered new.

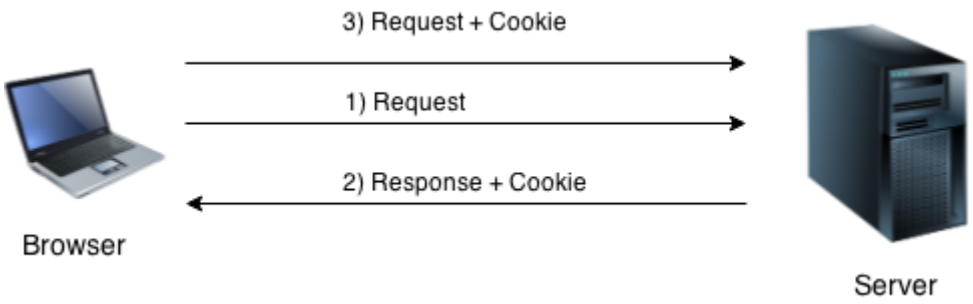


Fig : Flow of Cookies

Types of Cookie

There are 2 types of cookies in servlets.

1. **Non-persistent cookie**
   It is valid for single session only. It is removed each time when user closes the browser.
2. **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

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

Disadvantage of Cookies

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

Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

Constructor	**Description**
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	**Description**
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Other methods required for using Cookies

For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:

1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object.
2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser.

How to create Cookie?

- Cookie ck=new Cookie("user","sonoo jaiswal");//creating cookie object
- response.addCookie(ck);//adding cookie in the response

How to delete Cookie?

- Cookie ck=**new** Cookie("user","");//deleting value of cookie
- ck.setMaxAge(0);//changing the maximum age to 0 seconds
- response.addCookie(ck);//adding cookie in the response

How to get Cookies?

```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++){
 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
}
```

Example of Servlet Cookies

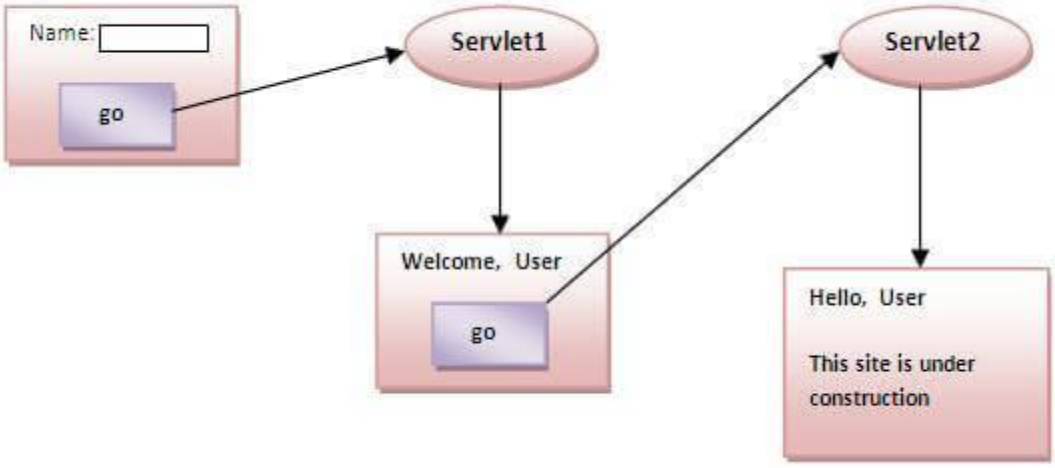


Fig : Output of Session Management Scenario

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.

Q. Explain Database Connectivity using JDBC.

1. **JDBC** stands for **Java Database Connectivity.**
2. **JDBC** is a **Java API** to connect and execute the query with the database.
3. It is a specification from Sun Microsystems that provides a standard abstraction(API or Protocol) for Java applications to communicate with various databases.
4. It provides the language with Java database connectivity standards.
5. It is used to write programs required to access databases.
6. JDBC, along with the database driver, can access databases and spreadsheets.

Architecture of JDBC

1. **Application:** It is a java applet or a servlet that communicates with a data source.
2. **JDBC API:** It provides various methods and interfaces for easy communication with the database. The JDBC API allows Java programs to execute SQL statements and retrieve results.
3. **JDBC Driver manager:** It loads a database-specific driver in an application to establish a connection with a database. It is used to make a database-specific call to the database to process the user request.
4. **JDBC drivers:** To communicate with a data source through JDBC, you need a JDBC driver that intelligently communicates with the respective data source.

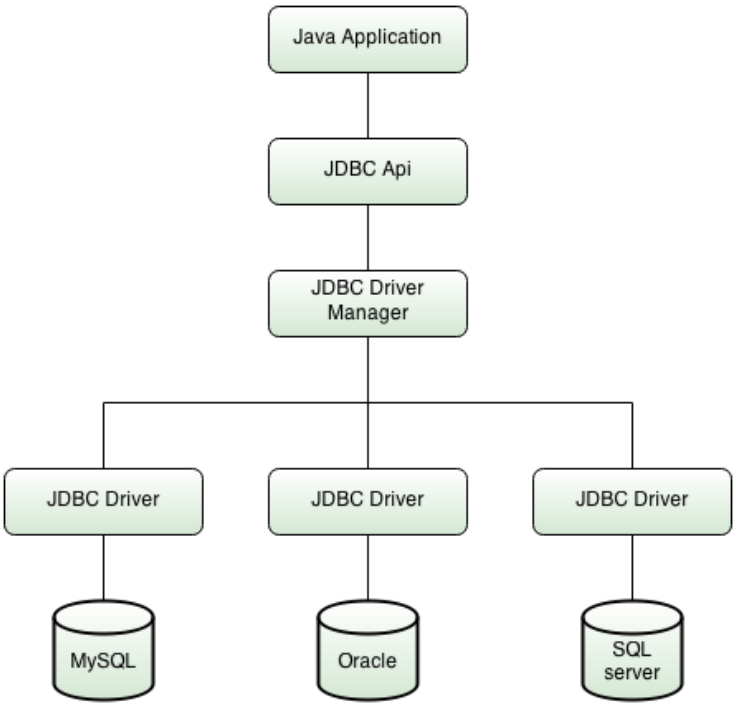


Fig : Architecture of JDBC

Steps in JDBC Workflow

1. Loading the Driver: The appropriate JDBC driver is loaded into memory using Class.forName() or by relying on Java's automatic service provider mechanism.
2. Establishing a Connection: A connection to the database is established using the Connection interface, obtained through the DriverManager.getConnection() method.

   Connection connection = DriverManager.getConnection(url, username, password);
3. Creating Statements: SQL queries are executed using either Statement for simple queries or PreparedStatement for parameterized queries.

   Statement statement = connection.createStatement();
4. Executing Queries: The executeQuery() method is used to execute SELECT queries, returning a ResultSet containing the query results.

   ResultSet resultSet = statement.executeQuery("SELECT * FROM users");
5. Retrieving and Modifying Data: Data from the ResultSet can be retrieved and manipulated using various methods provided by the interface.
6. Handling Exceptions: JDBC methods can throw SQLException instances, which must be handled to ensure graceful error management.
7. Closing Resources: It's essential to close Connection, Statement, and ResultSet objects to release database resources.

JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers:

1. Type-1 driver or JDBC-ODBC bridge driver
2. Type-2 driver or Native-API driver (partially java driver)
3. Type-3 driver or Network Protocol driver (fully java driver)
4. Type-4 driver or Thin driver (fully java driver)

Types of JDBC Architecture(2-tier and 3-tier)

The JDBC architecture consists of two-tier and three-tier processing models to access a database. They are as described below:

1. **Two-tier model:** A java application communicates directly to the data source. The JDBC driver enables the communication between the application and the data source. When a user sends a query to the data source, the answers for those queries are sent back to the user in the form of results.
   The data source can be located on a different machine on a network to which a user is connected. This is known as a **client/server configuration**, where the user's machine acts as a client, and the machine has the data source running acts as the server.

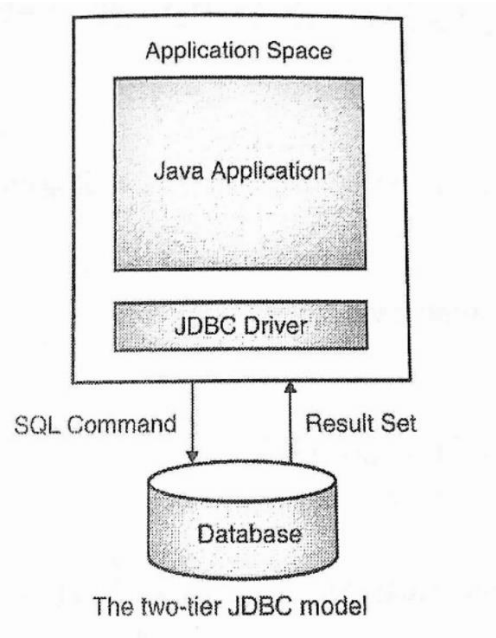


Fig : Two-tier JDBC model

2. **Three-tier model:** In this, the user's queries are sent to middle-tier services, from which the commands are again sent to the data source. The results are sent back to the middle tier, and from there to the user.

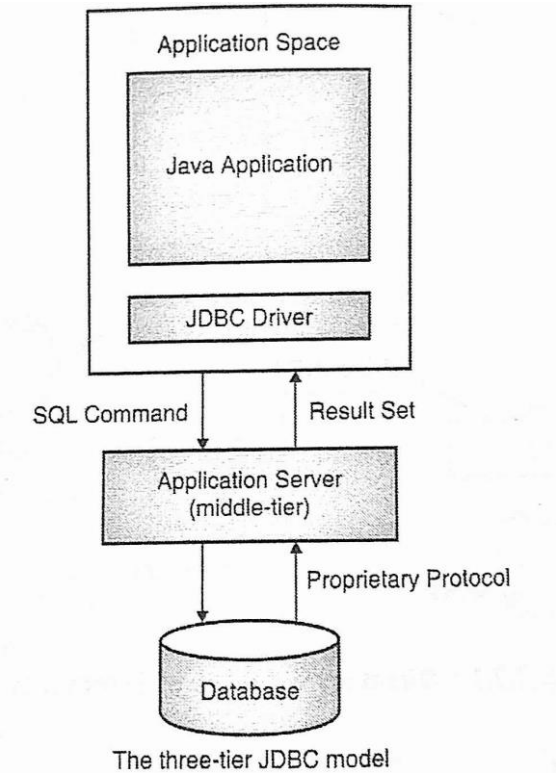


Fig : Three-tier JDBC model

Q. What is JSP (Java Server Page) ?

1. In Java, **JSP** stands **Java Server Pages**.
2. It is a server-side technology that is used for creating web applications.
3. It is used to create dynamic web content.
4. JSP consists of both HTML tags and JSP tags.
5. In this, JSP tags are used to insert JAVA code into HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
6. It is an advanced version of **Servlet** Technology i.e. a web-based technology that helps us to create dynamic and platform-independent web pages.
7. JSP is first converted into a servlet by the JSP container before processing the client's request.
8. JSP has various features like JSP Expressions, JSP tags, JSP Expression Language, etc.

Features of JSP

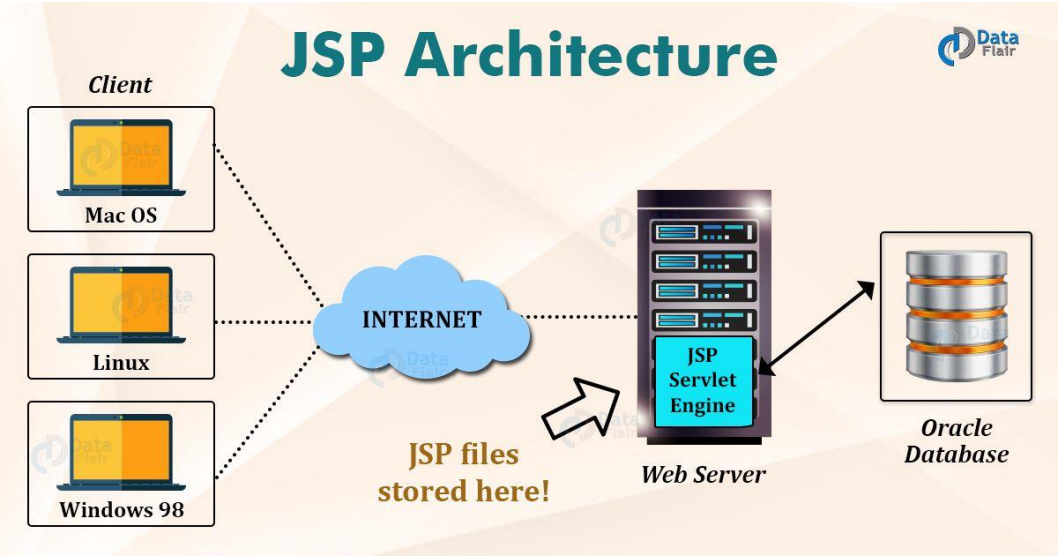
- **Coding in JSP is Easy:** As it involves adding Java code to HTML/XML.

- **Easy to Use and Learn:** It is straightforward and accessible for both Java and non-Java programmers.
- **It Does Not Require Advanced Knowledge of Java:** Suitable for users with basic Java skills.
- **Reduction in the Length of Code:** JSP uses action tags, custom tags, etc., to minimize code length.
- **JSP Expression:** Evaluates expressions and converts them to strings.
- **Declaration Tag:** Used to declare variables and methods within JSP pages.
- **Connection to Database is Easier:** Simplifies connecting to databases and allows for easy data reading and writing.
- **Extension to Servlet:** Inherits all features of servlets and includes implicit objects and custom tags.

Q. Explain 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.



Components of JSP architecture

- **JSP pages:** These are the main building blocks of a JSP application. They contain a combination of HTML, XML, and JSP elements (such as scriptlets, expressions, and directives) that generate dynamic content.

- **Servlets:** JSP pages are converted into servlets by the JSP engine. Servlets are Java classes that handle HTTP requests and generate dynamic content.
- **JSP engine (web container):** This web server component is responsible for processing JSP pages. It converts JSP pages into servlets, compiles them, and runs them in the Java Virtual Machine (JVM).

JSP Processing

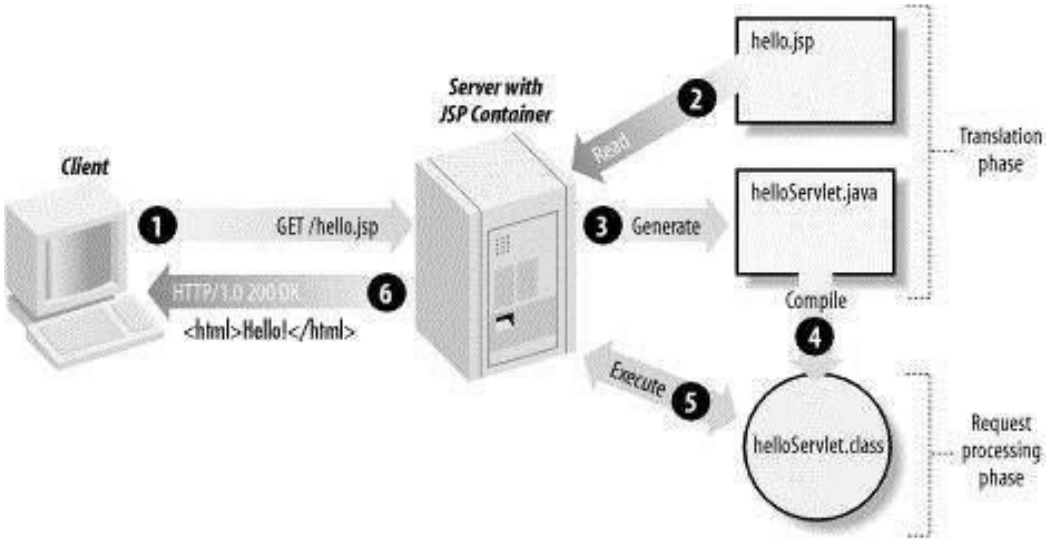


Fig : JSP Compilation Phase

1. A client (such as a web browser) sends a request for a JSP page to a web server.
2. The web server forwards the request to the JSP engine responsible for processing JSP pages.
3. The JSP engine checks if the requested JSP page has been compiled into a servlet. If not, it compiles the JSP page into a servlet class. This is done by parsing the JSP page and converting its elements (such as scriptlets, expressions, and directives) into Java code.
4. The JSP engine then compiles the servlet class, which creates a Java class file that can be executed by the Java Virtual Machine (JVM).
5. The JSP engine then creates an instance of the servlet class and calls the service() method, which generates the dynamic content for the JSP page. Within the service() method, the JSP engine generates the HTML code for the response by combining the static template in the JSP page with the dynamic content generated by the Java code.
6. The JSP engine sends the generated HTML code back to the web server, which then sends it back to the client as a response.
7. The JSP engine also maintains a cache of the compiled servlet classes so subsequent requests for the same JSP page can be handled more efficiently.

Q. Explain JSP Life Cycle

The JSP (JavaServer Pages) life cycle consists of several stages that govern the creation, execution, and destruction of a JSP page.

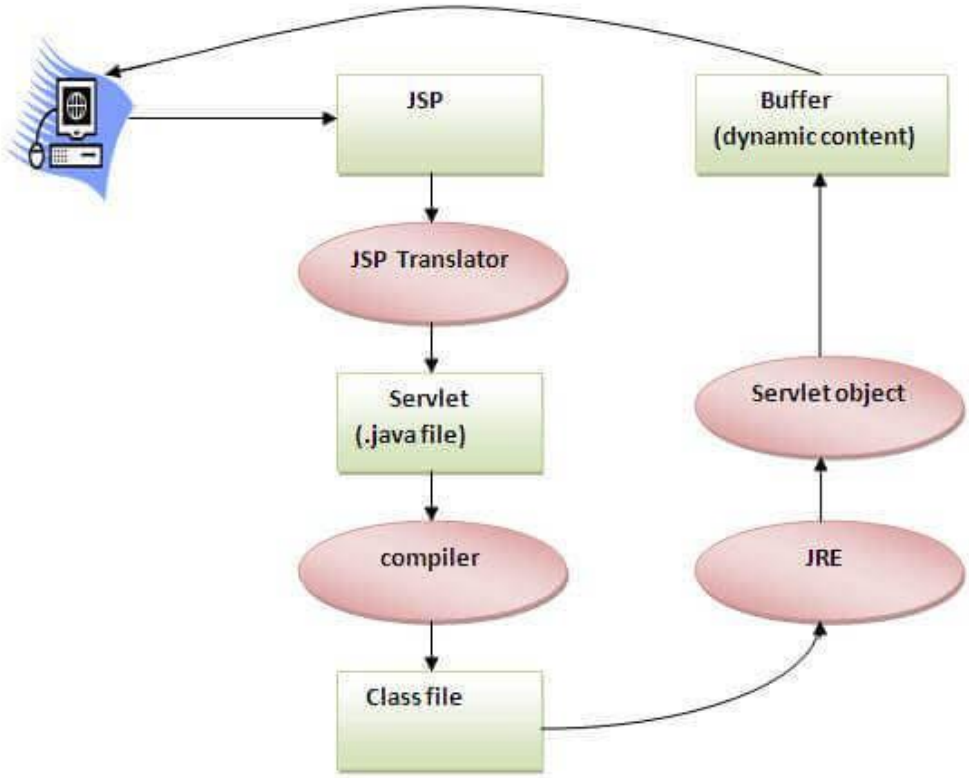


Fig : Life Cycle of JSP Page

1. **Translation Phase**:
    - When a JSP page is requested for the first time, the server translates it into a servlet. This involves parsing the JSP file, converting the JSP elements (like HTML, Java code, and JSP tags) into Java code.
2. **Compilation Phase**:
    - The generated servlet code is then compiled into bytecode by the Java compiler. This compiled servlet is stored for future requests.
3. **Loading Phase**:
    - The servlet class is loaded into memory when a request for the JSP page is made. The server creates an instance of the servlet.
4. **Initialization Phase**:
    - The init() method of the servlet is called. This is where the servlet is initialized, and resources can be set up.

5. **Request Handling Phase**:
    - The service() method is invoked to handle incoming requests. This method processes requests and generates responses. It can call doGet() or doPost() methods depending on the HTTP request type.
6. **Response Generation Phase**:
    - The JSP page generates HTML (or other content) as the response to the client. This includes executing any embedded Java code or JSP tags.
7. **Destruction Phase**:
    - When the servlet is no longer needed, the destroy() method is called. This allows for cleanup of resources.

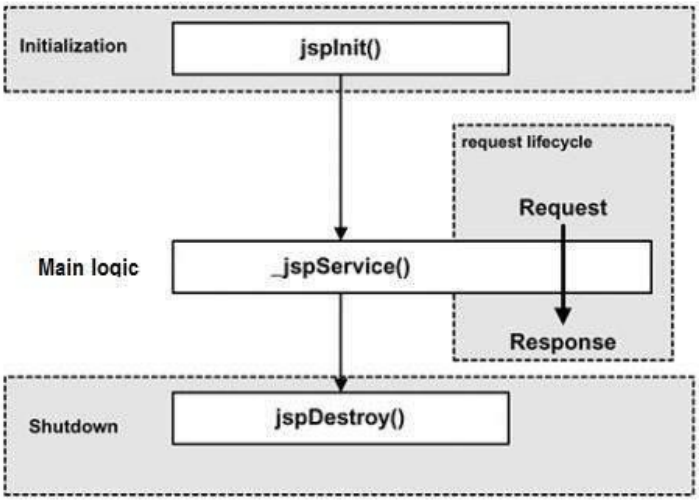


Fig : Generated Servlet Life Cycle

Q. Write a JSP Program to display date and time

```
<%@ page import="java.util.Date" %>
<!DOCTYPE html>
<html>
<head>
<title>Current Date and Time</title>
</head>
<body>
<h1>Current Date and Time</h1>
<% Date currentDate = new Date(); out.println("<p>" + currentDate.toString() + "</p>"); %>
</body>
</html>
```

Q. Write JSP Program to perform four basic arithmetic operation.

```
<!DOCTYPE html>
<html>
<head>
  <title>Arithmetic Operations</title>
</head>
<body>
  <h2>Arithmetic Operations</h2>
  <form action="calculate.jsp" method="post">
    <label for="num1">First Number:</label>
    <input type="text" id="num1" name="num1" required><br><br>
    <label for="num2">Second Number:</label>
    <input type="text" id="num2" name="num2" required><br><br>
    <label for="operation">Operation:</label>
    <select id="operation" name="operation">
      <option value="add">Addition</option>
      <option value="subtract">Subtraction</option>
      <option value="multiply">Multiplication</option>
      <option value="divide">Division</option>
    </select><br><br>
    <input type="submit" value="Calculate">
  </form>
</body>
</html>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
  <title>Calculation Result</title>
</head>
<body>
  <h2>Calculation Result</h2>
  <%
    String num1Str = request.getParameter("num1");
    String num2Str = request.getParameter("num2");
    String operation = request.getParameter("operation");

    double num1 = Double.parseDouble(num1Str);
    double num2 = Double.parseDouble(num2Str);
    double result = 0;
```

```
        switch (operation) {
            case "add":
                result = num1 + num2;
                break;
            case "subtract":
                result = num1 - num2;
                break;
            case "multiply":
                result = num1 * num2;
                break;
            case "divide":
                if (num2 != 0) {
                    result = num1 / num2;
                } else {
                    out.println("Error: Division by zero is not allowed.");
                    return;
                }
                break;
        }
        out.println("The result of " + operation + " is: " + result);
    %>
</body>
</html>
```