

# **Module 8) Web Technologies in Java**

**HTML Tags: Anchor, Form, Table, Image, List Tags, Paragraph, Break, Label**

## **1. Introduction to HTML and Its Structure**

**Ans: HTML (HyperText Markup Language) is the standard language used to create and design web pages. It describes the structure of a web page using tags.**

**HTML files are saved with the extension .html or .htm.**

### **Basic Structure of an HTML Document**

```
<!DOCTYPE html>

<html>
  <head>
    <title>My First Web Page</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is my first HTML page.</p>
  </body>
</html>
```

### **Explanation:**

- `<!DOCTYPE html>` → Defines HTML5 document
- `<html>` → Root element
- `<head>` → Contains meta information (title, styles, etc.)
- `<body>` → Contains visible content of the webpage

## **2. Explanation of Key HTML Tags**

## **1. <a> – Anchor Tag (Hyperlink)**

**Used to create hyperlinks to another webpage or location.**

**Syntax:**

```
<a href="https://www.google.com">Visit Google</a>
```

**Attributes:**

- **href → URL of the link**
- **target="\_blank" → Opens link in new tab**

## **2. <form> – Form Tag**

**Used to collect user input and send data to the server.**

**Example:**

```
<form action="submit.jsp" method="post">  
    <input type="text" name="username">  
    <input type="submit">  
</form>
```

**Common attributes:**

- **action → URL where data is sent**
- **method → GET or POST**

## **3. <table> – Table Tag**

**Used to display data in rows and columns.**

**Example:**

```
<table border="1">
```

```
<tr>
  <th>ID</th>
  <th>Name</th>
</tr>
<tr>
  <td>1</td>
  <td>Rahul</td>
</tr>
</table>
```

**Related tags:**

- **<tr>** → Table row
- **<th>** → Table heading
- **<td>** → Table data

#### 4. **<img>** – Image Tag

Used to display images on a web page.

**Example:**

```

```

**Attributes:**

- **src** → Image path
- **alt** → Alternate text
- **width, height** → Image size

#### 5. List Tags – **<ul>, <ol>, <li>**

**Unordered List (<ul>)**

```
<ul>  
  <li>HTML</li>  
  <li>CSS</li>  
  <li>JavaScript</li>  
</ul>
```

#### Ordered List (<ol>)

```
<ol>  
  <li>Login</li>  
  <li>Select Product</li>  
  <li>Payment</li>  
</ol>
```

- <li> → List item

## 6. <p> – Paragraph Tag

Used to define paragraphs of text.

Example:

```
<p>This is a paragraph.</p>
```

## 7. <br> – Line Break Tag

Used to break a line without starting a new paragraph.

Example:

```
Hello<br>World
```

It is a self-closing tag.

## 8. <label> – Label Tag

Used to define labels for form input elements.

**Example:**

```
<label for="name">Name:</label>  
<input type="text" id="name">
```

**Benefits:**

- **Improves form accessibility**
- **Clicking the label focuses the input field**

### ❖ **CSS: Inline CSS, Internal CSS, External CSS**

## 1. Overview of CSS and Its Importance in Web Design

**Ans:** CSS (Cascading Style Sheets) is used to control the appearance and layout of web pages. While HTML defines the structure of a webpage, CSS defines how the content looks.

**Importance of CSS:**

- **Improves website appearance**
- **Controls colors, fonts, spacing, and layout**
- **Makes web pages responsive**
- **Separates content (HTML) from design (CSS)**
- **Saves time by reusing styles across pages**

## 2. Types of CSS

There are three main types of CSS:

### 2.1 Inline CSS

**Inline CSS** is written directly inside an HTML element using the **style** attribute.

**Example:**

```
<p style="color: red; font-size: 18px;">
```

## This is Inline CSS

```
</p>
```

### Features:

- Applies style to a single element
- Has the highest priority
- Not reusable

### Advantages:

- Quick styling
- Useful for small changes

### Disadvantages:

- Difficult to maintain
- Makes HTML code messy

## 2.2 Internal CSS

Internal CSS is written inside a `<style>` tag within the `<head>` section of an HTML document.

### Example:

```
<!DOCTYPE html>

<html>
<head>
<style>
  p {
    color: blue;
    font-size: 16px;
  }
</style>
```

```
</head>  
<body>  
    <p>This is Internal CSS</p>  
</body>  
</html>
```

#### Features:

- Applies styles to one page
- Better than inline CSS for page-level styling

#### Advantages:

- Easy to manage for single pages
- Cleaner than inline CSS

#### Disadvantages:

- Not reusable across multiple pages

### 2.3 External CSS

External CSS is written in a separate .css file and linked to the HTML document using the `<link>` tag.

#### CSS File (style.css):

```
p {  
    color: green;  
    font-size: 20px;  
}
```

#### HTML File:

```
<link rel="stylesheet" href="style.css">
```

#### Features:

- Styles applied to multiple pages

- Best practice for large websites

**Advantages:**

- Easy maintenance
- Reusable styles
- Clean HTML code
- Faster page loading (cached)

**Disadvantages:**

- Requires an additional file

### **3. Priority Order of CSS**

**When multiple CSS types are applied, priority order is:**

- **Inline CSS (Highest)**
- **Internal CSS**
- **External CSS (Lowest)**

### **❖ CSS: Margin and Padding**

#### **➤ Theory:**

- **Margin:-** Margin is the space outside an element's border. It pushes the element away from other elements.
- **Padding:-** Padding is the space inside an element's border, between the border and the content.
- **Margin:-** Margins do not affect the element's background color, but padding does.
- Both margin and padding can be set for top, right, bottom, and left individually or together using shorthand.
- **Difference:** Margin = outside spacing, Padding = inside spacing.
- Using margin and padding helps in proper layout design and improves readability.

**Example:**

```
<!DOCTYPE html>

<html>
<head>
<style>

div {
    margin: 20px; /* space outside the div */
    padding: 15px; /* space inside the div */
    border: 2px solid blue;
    background-color: lightyellow;
}

</style>
</head>
<body>
<div>This is a div with margin and padding.</div>
</body>
</html>
```

### ❖ CSS: Pseudo-Class

**Theory:**

- **Pseudo-classes:-** Pseudo-classes in CSS allow you to style elements based on their state or position, without adding extra classes or IDs.
- Common pseudo-classes include :hover (when the mouse is over an element), :focus (when an element like input is selected), :active (when an element is being clicked), :first-child, :last-child, etc.

- They are written after the selector using a colon (:).
- Pseudo-classes make web pages interactive and dynamic without JavaScript.
- They help in changing styles for links, buttons, form fields, or list items depending on user interaction.

**Example:**

```
<!DOCTYPE html>

<html>
  <head>
    <style>
      a:hover {
        color: red; /* Changes link color when hovered */
      }

      input:focus {
        border: 2px solid green; /* Changes border when input is focused */
      }

      button:active {
        background-color: yellow; /* Changes button color when clicked */
      }
    </style>
  </head>
  <body>
    <a href="#">Hover over me</a><br><br>
    <input type="text" placeholder="Focus on me"><br><br>
    <button>Click me</button>
  </body>
```

</html>

### ❖ CSS: ID and Class Selectors

#### ➤ Theory:

- **Ans: An id is a unique identifier for a single HTML element, while a class can be assigned to multiple elements.**
- **In CSS, id selectors use # (e.g., #header), and class selectors use . (e.g., .menu).**
- **id is used when you want to style or target only one specific element.**
- **class is used when you want to apply the same style to multiple elements.**
- **IDs have higher specificity than classes in CSS, so they can override class styles.**
- **Using class promotes reusability and cleaner code, whereas id is ideal for unique elements like headers, footers, or a specific section.**

#### Example:

```
<!DOCTYPE html>

<html>
  <head>
    <style>
      #header {
        background-color: lightblue; /* Only one header will have this */
        text-align: center;
      }
      .menu {
        color: green;           /* All elements with class "menu" */
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is a sample web page.</p>
    <ul class="menu">
      <li>Link 1</li>
      <li>Link 2</li>
      <li>Link 3</li>
    </ul>
  </body>
</html>
```

```
font-weight: bold;  
}  
</style>  
</head>  
<body>  
  
<h1 id="header">Welcome to My Website</h1>  
<p class="menu">Home</p>  
<p class="menu">About</p>  
<p class="menu">Contact</p>  
  
</body>  
</html>
```

## ❖ Introduction to Client-Server Architecture

### 1. Overview of Client-Server Architecture:

**Ans:** Client-server architecture is a model that divides tasks between service providers, called servers, and service requesters, called clients. In this model, the client initiates a request for resources or services, and the server responds by providing the requested resources.

- Servers are typically powerful machines that store data, manage applications, and handle multiple client requests simultaneously. Clients can be computers, mobile devices, or applications that access the server over a network.
- The communication between client and server usually happens over a network, such as a local area network (LAN) or the Internet. This architecture supports centralized management, meaning updates and maintenance are easier to handle on the server side.
- It is widely used in web applications, email systems, and online banking platforms. For example, when you open a website, your

web browser (client) sends a request to the web server, which processes it and sends back the web page.

- This model enhances resource sharing, security, and scalability. Popular protocols used include HTTP, FTP, and SMTP. The design ensures clients are lightweight while servers handle heavy data processing.
- Client-server architecture also allows multiple clients to access the same server resources concurrently. It provides centralized control, efficient resource management, and easier maintenance.

## **2. Difference Between Client-Side and Server-Side Processing:**

**Ans:** Client-side and server-side processing refer to where the computation and processing take place in a client-server environment. Client-side processing occurs on the client's device, typically inside a web browser or application.

- Tasks like form validation, user interface updates, and interactive features are handled on the client side using languages like JavaScript. Server-side processing happens on the server, performing tasks such as database queries, user authentication, and application logic.
- Server-side languages include Java, PHP, Python, and Node.js. The main difference is that client-side reduces server load and provides faster response for interactive features, while server-side ensures data security, central control, and the ability to handle complex operations.
- For example, when a user submits a login form, client-side scripts might check if fields are empty, while the server-side script verifies the username and password against a database. Client-side processing is visible to users and can be modified, whereas server-side processing is hidden and more secure.
- Modern web applications often use a combination of both client-side and server-side processing for efficiency. Choosing where to process depends on performance, security, and application requirements.

### **3. Roles of a Client, Server, and Communication Protocols:**

**Ans:** In a client-server environment, the client, server, and communication protocols each have specific roles. The **client** is the requester of services, usually an application or device that interacts with the server, such as a web browser or email app.

- The **server** is the provider of services, hosting resources like web pages, databases, and applications, and it responds to client requests. Servers often handle multiple clients simultaneously and manage resources efficiently.
  - **Communication protocols** define the rules and standards for exchanging data between client and server, ensuring accurate and reliable transmission. Common protocols include HTTP/HTTPS for web pages, FTP for file transfer, and SMTP/IMAP for emails.
  - For example, when a user searches for a product on an e-commerce website, the client sends an HTTP request to the server. The server processes the request, fetches data from the database, and sends an HTTP response back to the client.
  - Protocols ensure both parties understand the format and method of communication. This setup allows distributed computing, secure transactions, and efficient resource sharing across networks.
- 
- **HTTP Protocol Overview with Request and Response Headers**

#### **1. Introduction to the HTTP Protocol and Its Role in Web Communication:**

**Ans:** HTTP (Hypertext Transfer Protocol) is the standard protocol used for communication between web clients and servers. It defines how messages are formatted, transmitted, and how web servers and browsers respond to requests.

- HTTP enables clients, such as web browsers, to request resources like HTML pages, images, and videos from servers. It is the foundation of data exchange on the World Wide Web and operates over TCP/IP.

## **2. Explanation of HTTP Request and Response Headers:**

**Ans:** HTTP request headers contain information sent by the client to the server, such as the type of browser, accepted content formats, and authentication details.

- HTTP response headers are sent by the server to the client, providing metadata like content type, server details, and status codes (e.g., 200 OK, 404 Not Found).
- These headers ensure proper communication, resource handling, and understanding between client and server during web interactions.

### **❖ J2EE Architecture Overview**

## **1. Introduction to J2EE and Its Multi-Tier Architecture:**

**Ans:** J2EE (Java 2 Platform, Enterprise Edition) is a platform designed for developing and deploying large-scale, distributed, and multi-tier web applications. It provides a set of services, APIs, and protocols that simplify enterprise-level application development.

- J2EE uses a **multi-tier architecture**, typically divided into three layers: the **client tier**, **business tier**, and **enterprise information tier**. The client tier handles user interfaces, the business tier processes application logic, and the enterprise tier manages data storage and retrieval.
- This separation of concerns improves scalability, maintainability, and flexibility in enterprise applications. For example, a banking application can have a web interface for customers, a middle layer for transaction processing, and a database server for account data.

## **2. Role of Web Containers, Application Servers, and Database Servers:**

**Ans:** A **web container** manages web components like servlets and JSPs, handling requests from clients and generating responses. It provides services such as session management and security.

- An **application server** hosts business components like EJBs (Enterprise JavaBeans) and manages business logic, transactions, and messaging. It ensures reliable execution of enterprise applications.
- A **database server** stores and manages data, providing secure access to application servers and ensuring data integrity. Together, these components support a robust, multi-tier J2EE environment for scalable and maintainable applications.

**❖ Web Component Development in Java (CGI Programming)**

### **1. Introduction to CGI (Common Gateway Interface):**

**Ans:** CGI (Common Gateway Interface) is a standard protocol that allows web servers to execute external programs, like scripts, to generate dynamic web content. It enables interaction between a web browser (client) and a web server.

#### **Process of CGI Programming:**

When a client sends a request, the server runs the CGI script, which processes the input, interacts with databases if needed, and sends back a response to the client.

#### **Advantages and Disadvantages:**

- Advantages of CGI include simplicity, language independence, and wide compatibility with web servers.
- Disadvantages are poor performance under heavy load and limited scalability, as each request creates a new process.

**❖ Servlet Programming: Introduction, Advantages, and Disadvantages**

### **1. What are servlets and how do they work?**

**Ans:** Servlets are Java programs that run on a web server to handle client

requests and generate dynamic web content. They operate within a servlet container, which manages the servlet's life cycle and provides services like session handling and request processing.

When a client sends a request, the servlet processes it, may interact with databases or other resources, and sends back a response, usually in HTML or JSON format. Servlets extend server capabilities beyond static pages.

## **2. What are the advantages and disadvantages of servlets?**

**Ans:** Advantages include high performance compared to CGI, portability across platforms, multi-threading, and secure handling of requests. Disadvantages are a steeper learning curve and the requirement of a servlet container to run.

### **❖ Servlet Versions and Types of Servlets**

#### **1. What is the history of servlet versions?**

**Ans:** Servlet 1.0 provided basic support for handling HTTP requests. Servlet 2.x introduced features like filters, listeners, and session management. Servlet 3.x added annotations, asynchronous processing, and better integration with Java EE.

#### **2. What are the types of servlets?**

**Ans:** There are two main types: **GenericServlet**, which is protocol-independent and requires overriding the **service()** method, and **HttpServlet**, which is HTTP-specific and provides built-in methods like **doGet()** and **doPost()** for handling web requests.

### **❖ Difference Between HTTP Servlet and GenericServlet**

#### **1. How do HTTP servlets differ from GenericServlet?**

**Ans:** GenericServlet is protocol-independent and uses the **service()** method to handle requests, so it can work with protocols other than HTTP.

- HttpServlet is HTTP-specific, providing methods like **doGet()**, **doPost()**, **doPut()**, and **doDelete()**, making it easier to handle standard web requests in web applications.

## ❖ Servlet Life Cycle

### 1. What are the phases of the servlet life cycle?

**Ans:** The servlet life cycle has three main phases. init() is called once when the servlet is loaded for initialization. service() is called for each client request to process and respond. destroy() is called once when the servlet is removed from memory for cleanup.

- The servlet container manages these phases automatically to ensure efficient resource management and request handling.

## ❖ Creating Servlets and Servlet Entry in web.xml

### 1. How do you create servlets and configure them using web.xml?

**Ans:** To create a servlet, you write a Java class that extends HttpServlet or GenericServlet and override required methods like **doGet()** or **service()**.

- After creating the servlet, it must be configured in web.xml, the deployment descriptor of a web application. In web.xml, you define a <servlet> tag with a <servlet-name> and <servlet-class>, and map it using <servlet-mapping> to assign a URL pattern.
- This configuration allows the servlet container to locate and invoke the servlet when a matching client request arrives.

## ❖ Logical URL and ServletConfig Interface

### 1. What are logical URLs and how are they used in servlets?

**Ans:** Logical URLs are the URL patterns defined in web.xml that map to a servlet, independent of the physical location of the servlet class. They help users access servlets using friendly or meaningful URLs.

### 2. What is the ServletConfig interface?

**Ans:** ServletConfig is an interface that provides configuration information for a servlet. Using methods like getInitParameter() and getServletName(), a servlet can access initialization parameters

defined in web.xml. This allows customization without changing the servlet code.

### ❖ RequestDispatcher Interface: forward() and include() Methods

#### 1. What is RequestDispatcher and how do forward() and include() methods work?

**Ans:** RequestDispatcher is an interface used to dispatch a request from one servlet to another resource (servlet, JSP, or HTML).

- forward() sends the request to another resource, replacing the current response.
- include() includes the content of another resource in the current response without replacing it.
- These methods are useful for modular applications, like forwarding login validation to another servlet or including a common header in multiple pages.

### ❖ ServletContext Interface and Web Application Listener

#### 1. What is ServletContext and what is its scope?

**Ans:** ServletContext is an interface that provides information about the web application and its environment. It has application-wide scope, meaning data stored here is accessible by all servlets in the application.

#### 2. How do web application listeners work?

**Ans:** Web application listeners monitor lifecycle events such as servlet context initialization, session creation, or attribute changes. They allow developers to perform actions when the application starts, shuts down, or when attributes are added, removed, or replaced.

### ❖ Java Filters: Introduction and Filter Life Cycle

## **1. What are filters in Java and when are they needed?**

**Ans:** Filters are Java components that intercept client requests and server responses in a web application. They are used for tasks such as authentication, logging, input validation, or data compression before a request reaches a servlet or after a response is generated.

## **2. What is the filter lifecycle and how are filters configured?**

**Ans:** The filter lifecycle includes three main methods: init(), called once when the filter is loaded; doFilter(), called for each request to process or modify requests/responses; and destroy(), called once when the filter is removed. Filters are configured in web.xml using <filter> and <filter-mapping> tags, specifying the filter class and URL patterns.

### **❖ Practical Example: Server-Side Validation Using Filters**

#### **1. How are filters used for server-side validation?**

**Ans:** Filters can check input fields before they reach a servlet. For example, a filter can inspect form parameters to ensure none are empty. If validation fails, the filter can forward the request back to the input form; otherwise, it allows the request to proceed to the servlet.

This approach centralizes validation logic, improves code reusability, and reduces repetitive checks in multiple servlets.

### **❖ JSP Basics: JSTL, Custom Tags, Scriptlets, and Implicit Objects**

#### **1. What are the key components of JSP?**

**Ans:** JSP (JavaServer Pages) allows dynamic content generation for web applications. Its key components include:

- **JSTL (JavaServer Pages Standard Tag Library):** Provides standard tags for iteration, conditionals, and formatting.
- **Custom Tags:** User-defined tags to encapsulate complex logic for reuse.

- **Scriptlets:** Java code embedded in JSP for dynamic content.
- **Implicit Objects:** Predefined objects like request, response, session, application for convenient access to web resources.

## ❖ Session Management and Cookies

### 1. What are session management techniques in Java web applications?

**Ans:** Session management is used to track user interactions across multiple requests. Techniques include cookies, hidden form fields, URL rewriting, and HttpSession objects.

### 2. How do web applications track user sessions?

**Ans:** Cookies store small data on the client's browser to identify sessions. Hidden form fields and URL rewriting pass session IDs in forms or URLs. The HttpSession object on the server maintains session data like user login status, preferences, and other contextual information, ensuring a consistent user experience.