**Introduction to JSP:-**

🡪**JSP** stands for JavaServer Pages.

🡪JSP is a technology used to create dynamic web pages by embedding Java code within HTML pages.

🡪JSP simplifies creating dynamic web content because we can write both HTML and Java code in the same file.

🡪Instead of writing complex Java code to output HTML (as with Servlets), we can write HTML directly and embed Java code only where we need it.

* **Features of JSP**
  + **Separation of Concerns**: JSP separates the user interface (HTML) from the business logic (Java code), making web development more organized.
  + **Ease of Use**: JSP allows embedding Java code within HTML using simple tags.
  + **JSP is built on top of Servlets**: When a JSP page is requested, it's compiled into a Servlet and then executed.
  + **Implicit Objects**: JSP provides many built-in objects (e.g., request, response, session) to simplify tasks like reading input, sending output, and managing sessions.
* **Basic Syntax of JSP**

🡪In a JSP page, HTML is written normally, but Java code can be embedded using special tags. These are:

1. **Directives**: These are instructions for the JSP engine and affect the entire JSP page. They set properties such as importing classes, configuring the page, including other files.

SYNTAX:- **<%@ ... %>**

Example: <%@ page import="java.util.Date" %>

🡪**Types of Directives in JSP:-**

* **Page Directives:-**Cong=figure page-level setting , like importing classes.

Example:- <%@ page import="java.util.Date" %>

* **Include Directives:-** Includes a file at the JSP compile time.

Example:- <%@ include file="hello.jsp" %>

🡪The include directive in JSP is used to include the content of one JSP file within another JSP file. Which reduces the code size by implementing the same jsp again and again also it helps to reusability of code and can update the code individually instead up as whole which enhance the scalability of that jsp file.

1. **Scriptlets**: Scriptlets allow to embed Java code within the JSP. Are often used for writing conditional logic or initializing variables.

**SYNATAX**:- **<% java variables declaration or condition establishment %>**

Example: <% int number = 10; %>

1. **Expressions**: Expressions evaluate Java code and directly display the result in the HTML output.

**SYNATAX**:- **<%= Expression %>**

Example:- <p>The result is: <%= 5 + 5 %></p>

It will give output as 10 and here we used <p> tag to show output on webpage

1. **JSP Declarations:** Declarations are used to declare variables or methods that are accessible throughout the JSP page.

**SYNTAX:-** **<%! Method or variables %>**

Example:- <%! int Square(int number) {

return number \* number;

} %>

1. **JSP Comments**: JSP Comments are not visible in the client’s HTML source. They’re used only for documentation and debugging in the JSP file.

**SYNTAX:- <%-- Comment message here--%>**

CODE EXAMPLE:- (First.jsp,header.jsp,footer.jsp) FILE\_Name:JSP\_01

**JSP API:-**

🡪The JSP API is a set of classes and interfaces, which enables the development of dynamic web content by embedding Java code within JSP.

* **Purpose of JSP API:-**
* **Dynamic Content Creation:** Embedding Java code directly into HTML to render dynamic content based on logic, conditions, or data from the server.
* **Servlet Integration**: Since JSP pages are converted into Java servlets during the compilation phase, they leverage the servlet API to interact with client requests and server responses.
* **Reusable Components**: Through directives, custom tags, and tag libraries, JSP enables reusable components across pages, enhancing maintainability and scalability.
* **Key Components of JSP API:-**

1. **JSP Directives:-**

* Page Directive
* Include Directive
* Taglib Directive

🡪The Taglib Directive in JSP allows to include reusable tags from a tag library, which provides many useful features, such as loops, conditionals, and formatting, without having to write much Java code.

**SYNTAX:- <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>**

* Explanation:-

**uri**: This is the unique identifier or address that tells JSP where to find the tag library.

**Prefix**: This is a short name (like c) that will use in JSP to call the tags from this library.

JSTL(JavaServer Pages Standard Tag Library) tags:-

* **Display Content** ->To display output. (**c:out**)
* **Set Variables with** ->Assigns values to variables in the page. (**c:set)**
* **If Condition** ->Displays content only if a condition is true. (**c:if**)
* **Loop** ->Repeats content for each item in a list. (**c:forEach)**
* **Condition->** It just work as if-else statement, allowing conditional branching.

**Note:-**

The test attribute in the <c:if> JSTL tag is a condition that evaluates to true or false. It is used to control whether the code inside the <c:if> block will execute.

CODE EXAMPLE:- (TAG\_lib.jsp) FILE\_Name:JSP\_02

1. **Implicit Objects:-**

JSP provides a set of predefined objects, known as implicit objects, which are automatically available within JSP pages. Some key implicit objects include:

* **request**: Represents the HttpServletRequest object, containing information about the client’s request.
* **response**: Represents the HttpServletResponse object, which is used to construct the response back to the client.
* **session**: Represents the HttpSession object, allowing data to persist across multiple client requests.
* **application**: Represents the ServletContext, which can store information accessible across the entire application.

NOTE:- Implicit methods contains the same method as Java servlets

1. **Scriptlets, Expressions, and Declarations**
2. **JSP API Lifecycle**
3. **Standard tag Library(JSTL)**

**JSP API Example:-**

CODE EXAMPLE:- JSP\_API.jsp FILE\_Name:JSP\_02

**JSP Request:-**

🡪The request is an instance of HttpServletRequest and represents the client's HTTP request, which can include data like form parameters, headers, cookies, and other request-related information.

* **Key Uses of the request:-**
* **Reading Request Parameters:** It can retrieve data sent by the client via form fields, query strings, or URL parameters.
* **Accessing Request Headers:** It allows reading HTTP headers, which may contain metadata like the client’s browser information.
* **Session Tracking:** It helps manage sessions by retrieving the associated HttpSession object.
* **Managing Attributes:** It supports adding or retrieving temporary attributes stored in the request scope.
* **Methods of Request:-**
* **getParameter(String name):** Retrieves the value of a request parameter as a String.
* **getParameterValues(String name):** Retrieves an array of String values for multi-valued parameters.
* **getHeader(String name):** Retrieves the value of a specific HTTP header.
* **getAttribute(String name) and setAttribute(String name, Object obj):** Manage custom attributes within the request scope.
* **getSession():** Retrieves the current HttpSession.

CODE EXAMPLE:- (index.html,Request.jsp) FILE\_Name:JSP\_02

**JSP Response:-**

🡪The response is an instance of HttpServletResponse and is responsible for sending data back to the client. Through this object, we can control the content type, redirect the user to another page, manage cookies, and set response headers.

* **Key Uses of the Response:-**
* **Setting Content Type:** Controls the format of the data sent to the client, like text/html or application/json.
* **Redirecting to Another URL:** Sends the client to a different page or resource.
* **Managing Cookies:** Adds cookies to the client’s browser.
* **Controlling HTTP Headers:** Sets HTTP response headers that may control caching, authentication, or custom behaviors.
* **Methods of Response:-**
* **setContentType(String type):** Specifies the MIME type of the response.
* **sendRedirect(String location):** Redirects the response to a different URL.
* **addCookie(Cookie cookie):** Adds a Cookie to the response.
* **setHeader(String name, String value):** Adds an HTTP header to the response.
* **getWriter():** Provides a PrintWriter to write text data directly to the client.

CODE EXAMPLE:- (login.html,login.jsp,welcome.jsp) FILE\_Name:JSP\_02

**JSP Session:-**

🡪The session is an instance of HttpSession, used to store information across multiple requests made by the same user. A session is created when a client first accesses the JSP and remains active until it times out or is invalidated.

* **Key Uses of the Sessions:-**
* Store data that needs to persist across multiple requests, like user authentication status, preferences, or shopping cart contents.
* Access data from previous requests during the user’s visit.
* **Methods of Session:-**
* **getAttribute(String name):** Retrieves an attribute stored in the session.
* **setAttribute(String name, Object value):** Sets or updates an attribute in the session.
* **removeAttribute(String name)**: Removes an attribute from the session.
* **invalidate():** Destroys the session, ending the current session for the user.

CODE EXAMPLE:- (index.jsp,session.jsp) FILE\_Name:JSP\_03

**JSP PageContext:-**

🡪The pageContext provides access to all available scopes and acts as a single point for accessing JSP-related data and resources on the page. It is an instance of PageContext.

* **Key Uses of the PageContext:-**
* Access and manipulate data in different scopes e.g.application, session, request, and page
* Access servlet-related information, attributes, and methods specific to the current page.
* **Methods of PageContext:-**
* **getAttribute(String name, int scope)**: Retrieves an attribute from a specific scope.
* **setAttribute(String name, Object value, int scope)**: Sets or updates an attribute in a specific scope.
* **removeAttribute(String name, int scope)**: Removes an attribute from a specific scope.
* **getSession()**: Returns the session associated with the page.
* **getRequest()**: Returns the request object for the current request.
* **getResponse()**: Returns the response object for the current request

CODE EXAMPLE:- (index1.jsp,pagecontext.jsp) FILE\_Name:JSP\_03

**JSP Exception:-**

🡪The exception object is used in JSP error pages to handle exceptions that occur during the request processing. It is available only in JSP pages where the **isErrorPage="true"** directive is set.

* **Key Uses of the JSP Exception:-**
* Retrieve information about exceptions that occur during page processing, enabling custom error handling.
* Display error details for debugging or logging purposes.
* **Methods of JSP Exception:-**
* **getMessage():** Returns a message about the error.
* **printStackTrace():** Prints the stack trace of the error.
* **getCause():** Returns the cause of the error, if any.

CODE EXAMPLE:- (ExceptionHandling.jsp,error.jsp) FILE\_Name:JSP\_03

**JSP Action Elements:-**

These are the special tags that control the flow of execution or the way content is included within a JSP page. Two of the most commonly used action elements are **<jsp:forward>** and **<jsp:include>.**

* **JSP forward tag:-**

🡪The **<jsp:forward>** action element is used to forward the request from one JSP page to another.

🡪When a request is forwarded, the target JSP or servlet processes the request as if it had been directly invoked by the client.

🡪This is useful for maintaining a clean separation between different parts of a web application, such as handling requests, displaying views, or redirecting users based on specific conditions.

* **Key Features of <jsp:forward>:**
* It transfers control to another resource e.g. JSP, servlet.
* The original request and response objects are preserved in the forwarded request.
* The client does not see a change in the URL in the browser; the forwarding happens on the server side.

**SYNTAX:- <jsp:forward page="Target\_page.extension" />**

CODE EXAMPLE:- (forward.jsp,target.jsp) FILE\_Name:JSP\_03

* **JSP include tag:-**

🡪The <jsp:include> action element allows to include another resource (JSP, servlet, or HTML file) within the current JSP page at runtime.

🡪This is useful for modularizing content, such as headers, footers, or common UI components, and dynamically including them in multiple JSPs without duplicating code.

* **Key Features of <jsp:include>:**
* It includes the content of the specified resource at the time the page is requested.
* The included resource can be a JSP, servlet, or static HTML file.
* The request and response objects are shared between the including and included resources.

**SYNTAX:- <jsp:include page="included\_Page.extension" />**

CODE EXAMPLE:- (include.jsp,header.html,footer.html) FILE\_Name:JSP\_03