Coding Bootcamp

TECHNOLOGY

JSP

TECHNOLOGY

Getting Started with JSP

simplilearn

Learning Objectives

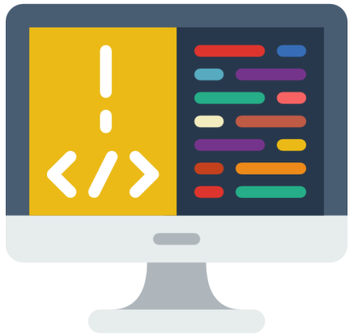
By the end of this lesson, you will be able to:

- Analyze the importance of using JSP to create robust, maintainable, and scalable web applications
- Identify the different JSP tags to standardize development practices and simplify complex operations
- Define the usage of JSP directive elements to manage the behavior and configuration of JSP pages
- Illustrate the use of action elements and tags to improve code readability and promote standardized practices

What Is JSP?

What Is JSP?

JSP is a server-side technology that allows you to create dynamic, platform-independent web applications.



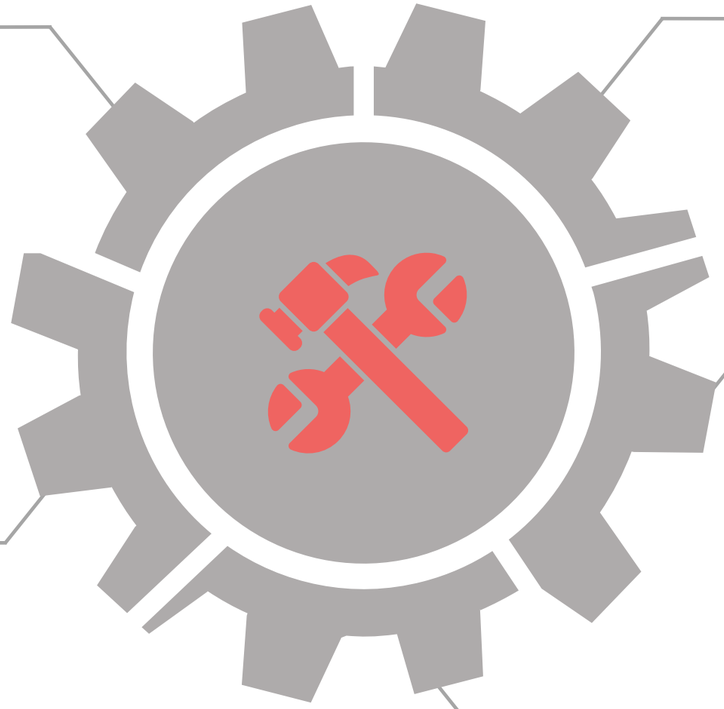
- JSP allows developers to embed Java code directly into HTML pages using special JSP tags, making it easier to create dynamic content.
- It generates content based on user input or other dynamic sources.

What Is JSP?

JSP requests are handled by the WebLogic server when:

The generated JSP page servlet class is invoked to handle the browser request.

The browser requests a page.



The WebLogic server refers to a JSP request.

The servlet class is triggered to handle the browser request.

The JSP compiler converts the JSP into a servlet class.

Integration with Java Enterprise Edition (EE)

JSP is part of the Java EE (Enterprise Edition) platform, which integrates seamlessly with other Java technologies like Servlets, JDBC, and EJBs.

Developers can build powerful, robust, and scalable web applications that meet enterprise-level requirements by integrating JSP with Java EE.

Integration with Java Enterprise Edition (EE)

Here is an overview of how JSP fits within the Java EE ecosystem:

JSP and Servlets

When a JSP page is requested, the web container compiles it into a servlet, allowing JSP to leverage all servlet features like request handling and lifecycle management.

JavaBeans

JavaBeans are reusable software components that are visually manipulated in builder tools. In JSP, they help separate business logic from presentation logic.

Why Use JSP?

JavaServer Pages (JSP) is a technology used for web development, and its main purposes include:

Dynamic web content creation	JSP embeds Java code in HTML for dynamic page generation.
Simplification of servlets	It converts HTML with embedded Java into servlets automatically.
Separation of concerns	It keeps presentation (HTML) and business logic (Java) separate for cleaner code.
Integration with Java EE	It works well with Java EE technologies for robust and scalable web applications.

Where to Use JSP?

Following are some essential uses of JSP

E-commerce sites: JSP dynamically generates web pages based on user actions, searches, and transactions.

Enterprise web apps: It is used in large enterprise apps for robust, scalable solutions that integrate well with Java EE components.

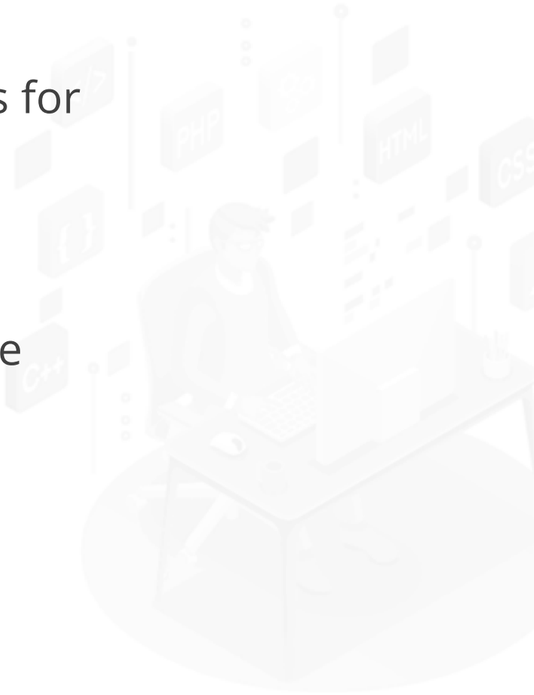
Content management systems (CMS): It manages and delivers dynamic content, easily integrating with databases and back-end systems.

Where to Use JSP?

Following are some essential uses of JSP

- **Customer relationship management (CRM):** It creates dynamic interfaces for CRM systems, improving user interaction and data presentation.

- **Educational platforms:** It helps to build interactive and personalized online learning experiences that integrate with various back-end services.



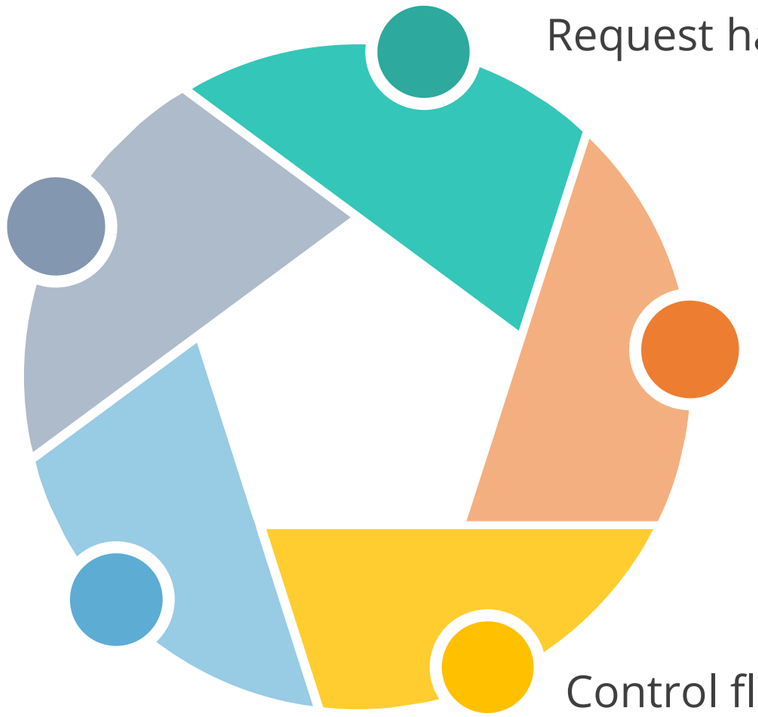
Servlets

Servlets: Introduction

Servlets are a type of Java class that runs on a Java-enabled server. They are server-side programs designed to handle client requests and generate dynamic web content.

An HTTP servlet is a specific servlet that processes HTTP requests and generates HTTP responses. This interaction is typically in HTML pages, making them essential for web applications.

Servlets: Uses



JavaBean

JavaBean: Introduction

JavaBeans are fundamental building blocks in Java applications, promoting encapsulation, reusability, and ease of use.

A JavaBean is a Java class that should follow these conventions:

1. It should have a no-arg constructor.
2. It must implement Serializable interface.
3. All the properties should be private, with public getter and setter methods.

JavaBean: Example

Example:

```
package techABC;
public class Item implements java.io.Serializable
{
private int id;
private String name;
public Item ()
{}
public void setItemId(int id)
{
this.id=id;}
public int getItemId()
{
return id;}
public void setItemName(String name)
{
this.name=name;}
public String getItemName()
{
return name;}}
```

Creating JSP Page

Problem Statement:

You have been asked to create a JSP (Java Server Pages) page using Eclipse IDE.

Outcome:

By leveraging JDBC's CallableStatement API to execute stored procedures, you'll master encapsulating complex database operations for reusability and efficiency. This approach boosts modularity and security, elevating overall application performance and maintainability.

Note: Refer to the demo document for detailed steps: 01_Create_JSP_Page

ASSISTED PRACTICE

Assisted Practice: Guidelines

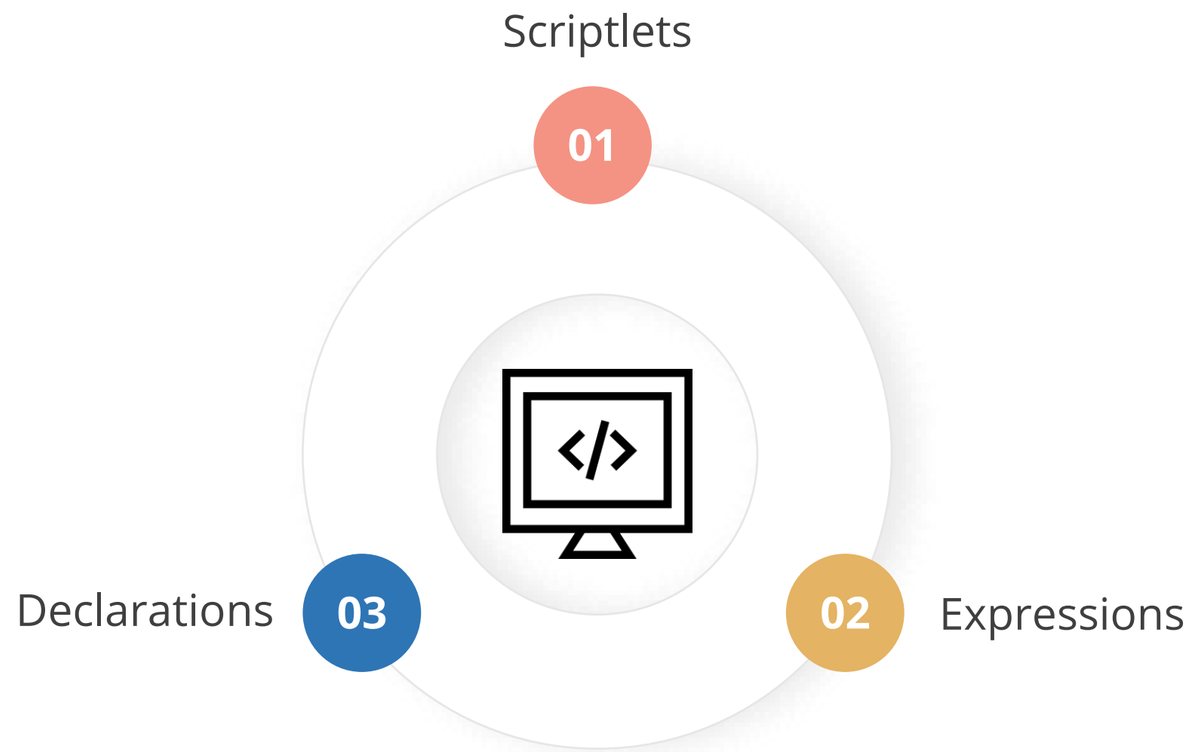
Steps to be followed are:

1. Create a new dynamic web project
2. Create a new JSP file
3. Incorporate changes in the previous operation
4. Create a new Maven project
5. Add the dependencies
6. Create a scriptlet tag

JSP Scripting Elements

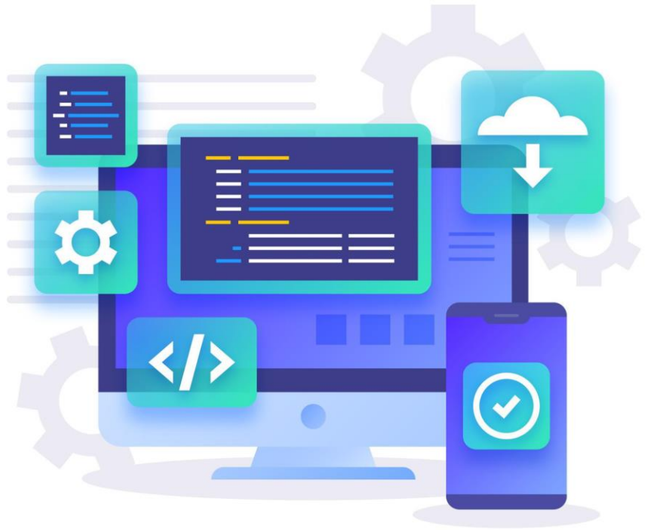
JSP Scripting Elements: Introduction

JSP scripting elements enable developers to embed Java code directly into HTML pages. There are three types of scripting elements in JSP:



Scriptlet Element

Scriptlets embed blocks of Java code within a JSP page. The code inside scriptlets is executed each time the JSP page is requested.



Syntax:

```
<%
    // Java code here
%>
```

Expression Element

Expressions output the result of a Java expression directly into the HTML output. The expression is evaluated, converted to a string, and inserted into the HTML.

Syntax:

```
<%= expression %>
```

Declaration Element

Declarations declare instance variables and methods that can be used in the JSP page.

The code inside declarations is placed outside the service method of the generated servlet, making it available to all the service methods.

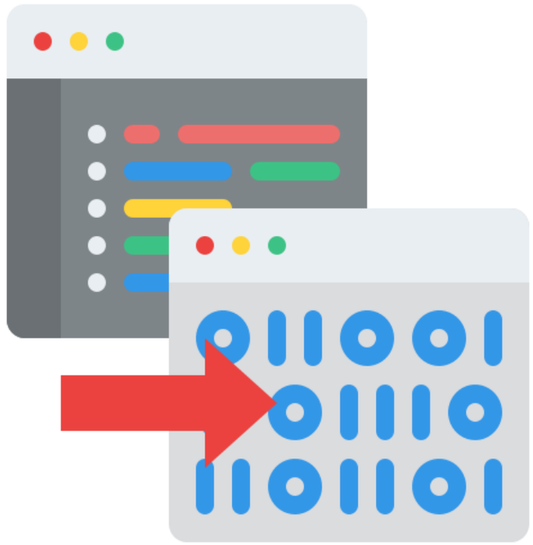
Syntax:

```
<%!
    // Variable or method
declaration here
%>
```

JSP Directive Elements

JSP Directive Elements

JSP directive elements provide global information about an entire JSP page and control the overall structure and behavior of the servlet generated from the JSP.



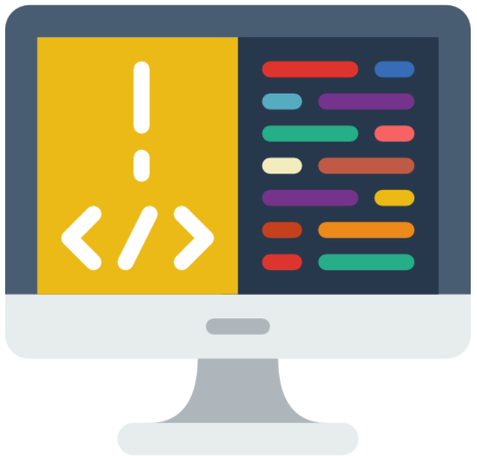
JSP Directive Elements

There are three types of directive elements in JSP:

1. Page directive
2. Include directive
3. Taglib directive

Page Directive

The page directive defines attributes that apply to the JSP page, such as character encoding, scripting language, error handling, and more.

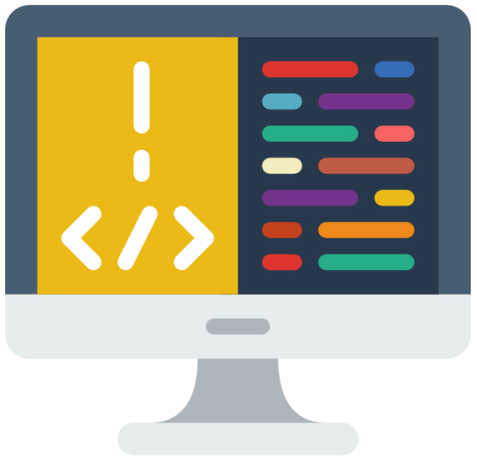


Syntax:

```
<%@ page attribute="value" %>
```

Include Directive

The include directive includes a static or JSP file during the translation phase.

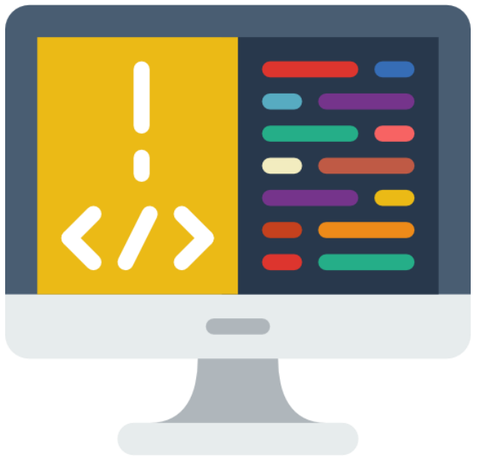


Syntax:

```
<%@ include file="relativeURL" %>
```

Taglib Directive

The taglib directive declares a custom tag library, making the tags defined in the library available to the JSP page.



Syntax:

```
<%@ taglib uri="uri"
prefix="prefix" %>
```

JSP Action Elements or Tags

JSP Action Elements

JSP action elements are XML-like tags that control the behavior of the servlet engine.

They perform specific tasks such as creating JavaBeans, including other resources, and forwarding requests.

JSP Action Elements: Uses

For managing JavaBeans

For forwarding requests

For passing the parameters

JSP Action Tags

JSP action tags are XML-based tags used to perform various tasks on a JSP page.

Here are the different action tags:

Tags	Description
jsp:forward	Forwards the request and response to another resource like servlet, JSP, and so on
jsp:include	Includes another resource
jsp:useBean	Creates or finds bean objects
jsp:setProperty	Sets the value of the bean object property
jsp:getProperty	Prints the value of the bean object property
jsp:plugin	Embeds other applications such as applets
jsp:param	Sets the parameter value and is used in forward and include
jsp:text	Writes the template text in JSP pages and documents

JSP Action Tags: Syntax

The syntax of action tags may be JSP, HTML, or another resource.

Syntax:

```
<jsp:forward page="relativeURL | <%=
expression %>" />
```

JSP Action Tags

The syntax of the **jsp:include** action tag may be JSP, HTML, or servlet.

Syntax:

```
<jsp:include page="relativeURL |
<%= expression %>" />
```

JSP Action Tags

Example:

index.jsp:

```
<html>
<body>
<h2>this is index page</h2>
<jsp:include page="printtime.jsp" />
<h2>end section of index page</h2>
</body>
</html>
```

JSP Action Tags

Example:

printtime.jsp:

```
<html>
<body>
<% out.print ("Today
is:"+java.util.Calendar.getInstance().get
Time()); %>
</body>
</html>
```

JSP Action Tags

The **jsp:useBean** is used to load a bean.

Syntax:

```
<jsp:useBean page = "name"/>
```

JSP Action Tags

Example:

```
<html>
<body>
<jsp:useBean page="printtime.jsp" />
</body>
</html>
```

JSP Action Tags

Example:

printtime.jsp:

```
<html>
<body>
<% out.print ("Today
is:"+java.util.Calendar.getInstance().getTim
e()); %>
</body>
</html>
```

JSP Action Tags

The syntax of the **jsp:setProperty** action tag may be JSP, HTML, or servlet.

Syntax:

```
<jsp:setproperty name=""
property="" >
```

JSP Action Tags

Example:

```
<html>
<body>
<jsp:setProperty name="printtime"
, property=”time”/>
</body>
</html>
```

JSP Action Tags

The syntax of the **jsp:getAttribute** action tag may be JSP, HTML, or servlet.

Syntax:

```
<jsp:getAttribute name=""
property="" >
```

JSP Action Tags

Example:

Syntax:

```
<html>
<body>
<jsp:getProperty name="printtime"
, property=”time”/>
</body>
</html>
```

JSP Action Tags

The **jsp:plugin** action tag introduces Java components into JSP. These components can either be an applet or a bean.

Syntax:

```
<jsp:plugin type="applet/bean"
code="objectcode"
codebase="objectcodebase">
```

JSP Action Tags

Example:

```
<html>
<body>
<jsp:plugin type="applet"
code=”test.employee”
codebase=”test.office”/>
</body>
</html>
```

JSP Action Tags

The **jsp:param** action tag is the child object of the plugin object.

Syntax:

```
<jsp:params>
<jsp:param name="val" value="val"/
>
</jsp:params>
```

JSP Action Tags

Example:

Syntax:

```
<html>
<body>
<jsp:plugin type="applet"  code="test.employee"
codebase="test.office"/>
<jsp:params>
     <jsp:param name="empcode" value="SL234" />
     <jsp:param name="name" value="Jack" />
  </jsp:params>
</body>
</html>
```

JSP Action Tags

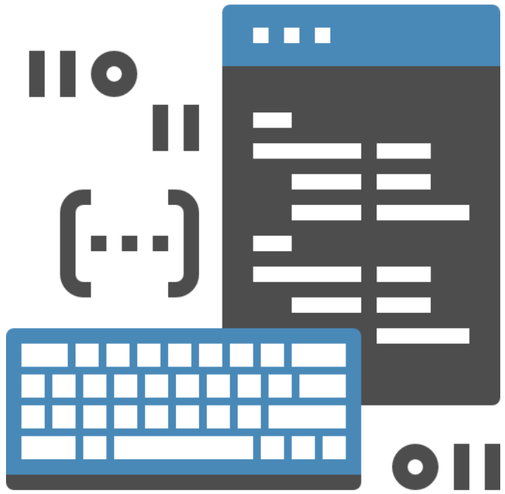
The **jsp:text** action tag templatizes JSP pages.

Syntax:

```
<jsp:text>template text</jsp:text>
```

JSP Action Tags

Example:



```
<html>
<body>
<jsp:text> This is Template
text</jsp:text>
</body>
</html>
```

JSP Action Tags

Problem Statement:

Using the Eclipse IDE, you have been asked to explore various JSP tags in a web application.

Outcome:

By exploring various JSP tags within a web application using the Eclipse IDE, you'll gain proficiency in dynamically generating content and controlling page behavior. This hands-on experience enhances your ability to create interactive and dynamic web applications efficiently.

Note: Refer to the demo document for detailed steps: 02_JSP_Action_Tags

ASSISTED PRACTICE

Assisted Practice: Guidelines

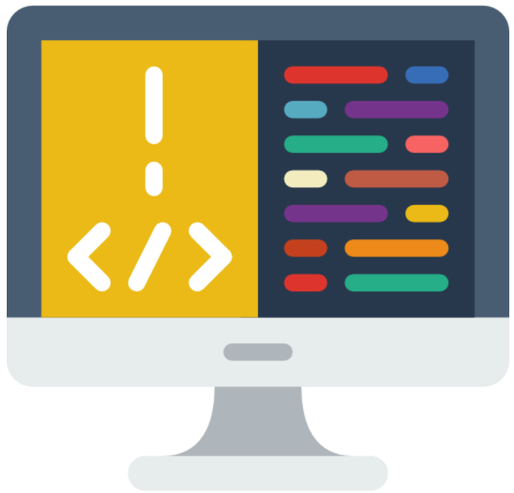
Steps to be followed are:

1. Create a new JSP File
2. Display the username on the webpage
3. Add a header to the **index.jsp** file
4. Create a new class
5. Create another class

Creating Custom JSP Tags

Custom JSP Tags: Overview

Custom JSP tags are user-defined tags that encapsulate reusable pieces of functionality in JSP pages.



They improve JSP pages' readability, maintainability, and modularity by abstracting complex operations into simple tag-based syntax.

Custom JSP Tags: Components

Tag handler class

Tag Library Descriptor (TLD) file

JSP page

Creating Custom JSP Tags

Here are the steps involved in creating and using custom JSP tags:

Tag handler class

A tag handler class is a Java class that implements the logic for the custom tag. It can extend SimpleTagSupport or implement interfaces like Tag, IterationTag, or BodyTag.

Tag library descriptor file

A TLD file maps the custom tag to its handler class and provides metadata about the tag.

Custom tags usage in JSP

To use custom tags in a JSP page, you need to declare the tag library and then use the custom tags as defined in the TLD file.

Custom JSP Tags: Benefits

1. Reusability
2. Readability
3. Maintainability
4. Modularity

Custom JSP Tags: Benefits

Reusability

Custom tags encapsulate reusable logic, making it easy to reuse across different JSP pages.

Readability

They simplify JSP pages by replacing complex Java code with simple, readable tags.

Maintainability

Encapsulating logic in tags makes it easier to manage and update the application.

Modularity

They promote a modular design, allowing developers to manage different functionalities separately.

JSP Tags

Problem Statement:

Using the Eclipse IDE, you have been asked to explore different JSP tags in a web application.

Outcome:

Exploring various JSP tags in a web application using the Eclipse IDE allows you to understand dynamic content generation and page control. This hands-on experience enhances your ability to create interactive and dynamic web applications effectively.

Note: Refer to the demo document for detailed steps: 03_JSP_Tags

ASSISTED PRACTICE

Assisted Practice: Guidelines

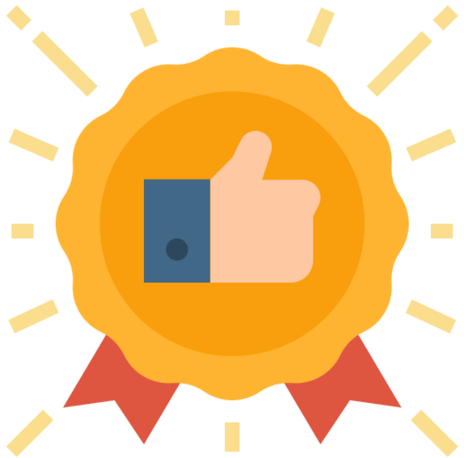
Steps to be followed are:

1. Create a new JSP file
2. Create an anchor tag to link tags.jsp file
3. Create a declarative tag
4. Copy the tags.jsp file to the eStoreMavenWeb project

Custom Tag Library

Custom Tag Library: Overview

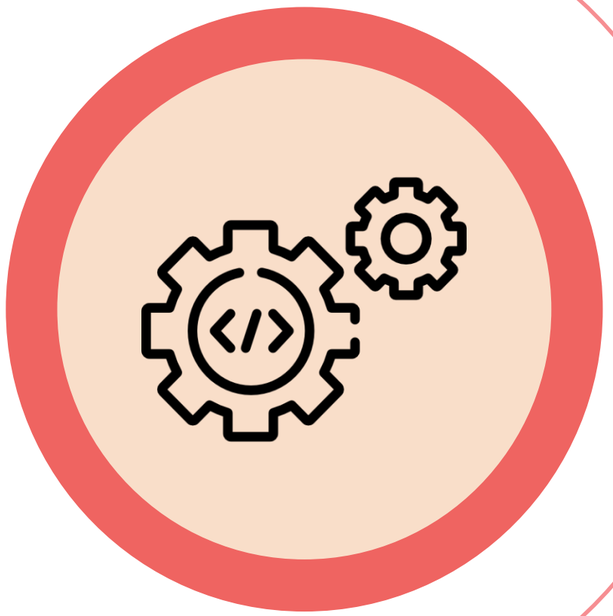
A custom tag library in JSP allows developers to define their custom tags, which can encapsulate complex functionality and promote reusability, modularity, and maintainability in JSP applications.



- It encapsulates reusable pieces of functionality, allowing them to be used across multiple JSP pages without code duplication.
- It promotes a modular approach to web application development by isolating specific functionalities into separate components.

Custom Tag Library: Functions

The custom tag library contains a lot of functions, such as:



- Is customized in terms of attributes passed from the calling JSP page
- Returns the variables to the calling page
- Accesses all the objects available on the JSP pages
- Accesses each other by creating and initializing a JavaBeans component
- Adds another page and communicate utilizing private variables

Benefits of Custom Tag Libraries

01 They can significantly enhance the development of dynamic and scalable web applications.

02 They abstract complex functionality into simple tags, making it easier for non-programmers to work with JSP pages.

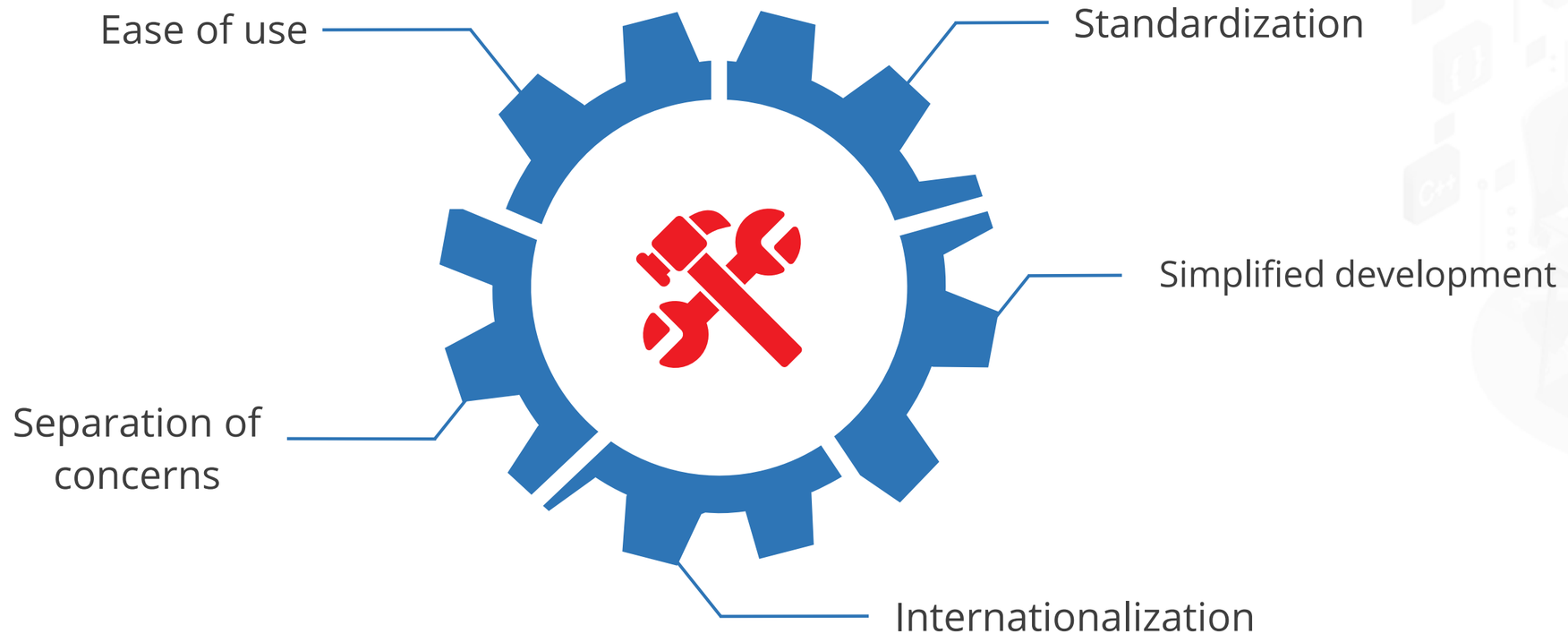
03 They provide extensive functionality that enhances the development of dynamic, modular, and maintainable web applications.

JSP Standard Tag Library

JSP Standard Tag Library

The JSP Standard Tag Library (JSTL) is a collection of custom tags encapsulating core functionality standards for many JSP applications.

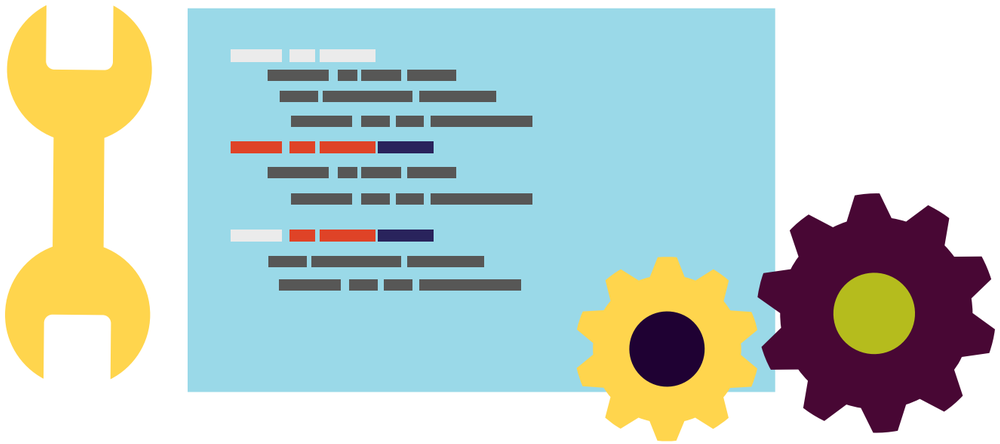
Here are the benefits of using JSTL:



JSTL Core Tags

JSTL Core Tags

The JSP Standard Tag Library (JSTL) core tags library provides essential functionality that simplifies the development of JSP-based web applications.



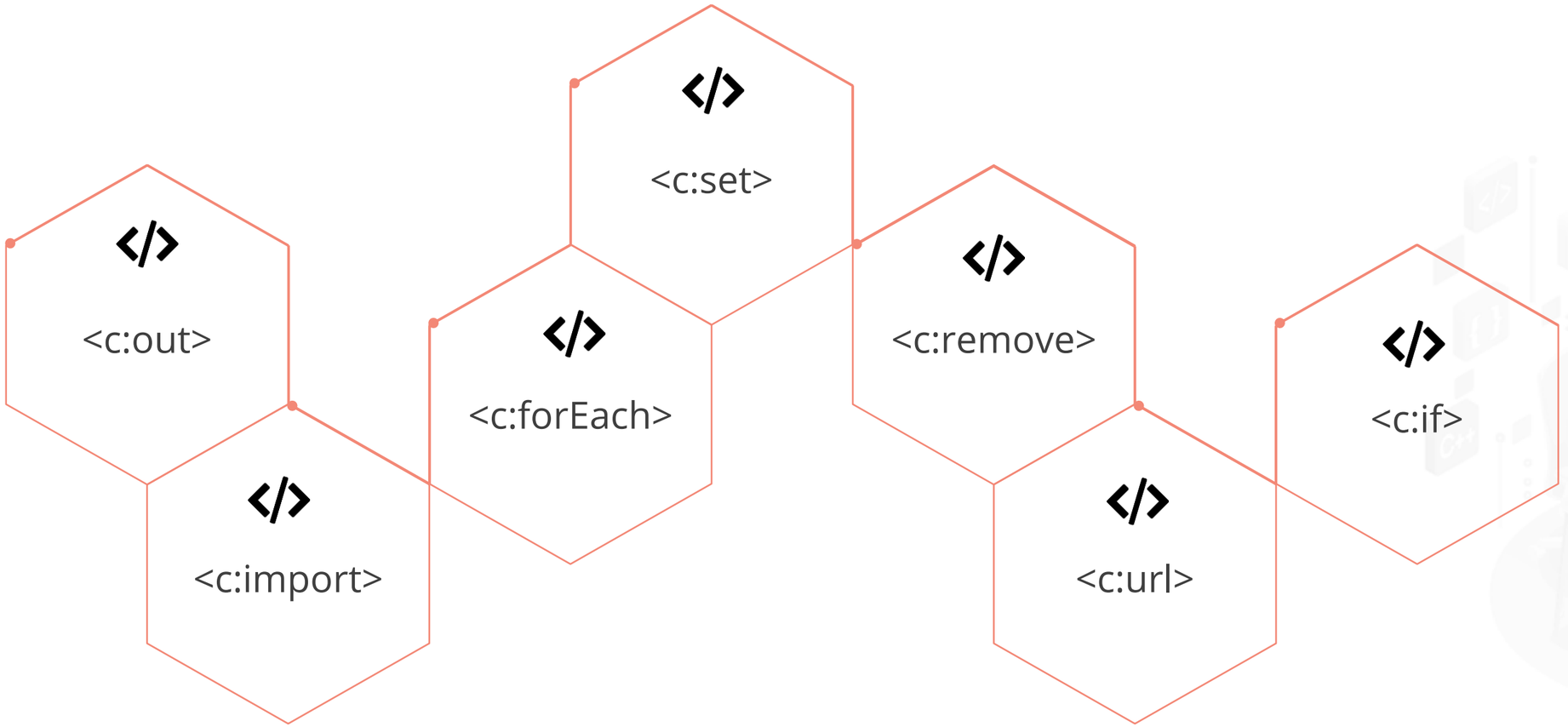
These tags handle common tasks such as iteration, conditionals, variable manipulation, URL management, and more.

Core Tags: Features

Here are the key features of core tags:

- Simplify the process of looping through collections and handling conditional logic directly in JSP pages
- Allows the setting and retrieval of variables in various scopes (page, request, session, application)
- Assist in managing URLs for redirection and parameter handling
- Provide support for tasks like removing variables, catching exceptions, and handling out-of-context expressions

Core Tags: Types



The <c:out> Tag

It outputs the value of an expression, escaping XML characters to prevent XSS attacks.

Syntax:

```
<c:out value="expression"
[default="defaultValue"]
[escapeXml="true|false"] />
```

The <c:import> Tag

It imports the content of a URL and includes it in the JSP page.

Syntax:

```
<c:import url="url" [var="varName"]
[scope="page|request|session|application"] />
```

The <c:set> Tag

It sets the value of a variable in a specified scope.

Syntax:

```
<c:set var="varName" value="expression"
[scope="page|request|session|application"] />
```

The <c:forEach> Tag

It iterates over a collection of items.

Syntax:

```
<c:forEach var="item" items="collection"
[begin="start"] [end="end"] [step="step"]
[varStatus="statusVar"]>
    <!-- Body content -->
</c:forEach>
```

The <c:remove> Tag

It removes a variable from a specified scope.

Syntax:

```
<c:remove var="varName"
[scope="page|request|session|application"] />
```

The <c:url> Tag

It constructs a URL with optional query parameters.

Syntax:

```
<c:url value="url" var="varName">
    <c:param name="paramName" value="paramValue" />
</c:url>
```

The <c:if> Tag

It conditionally evaluates its body content if the test expression is true.

Syntax:

```
<c:if test="expression">
    <!-- Body content -->
</c:if>
```

Exception Handling in JSP

Problem Statement:

You have been asked to explore exception handling in JSP.

Outcome:

By exploring exception handling in JSP, you'll understand how to gracefully manage errors and exceptions that occur during web application execution. This knowledge empowers you to create robust and reliable web applications that handle unexpected issues without compromising user experience.

Note: Refer to the demo document for detailed steps: 04_Exception_Handling_in_JSP

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Open the tags.jsp file
2. Create a new JSP File
3. Create another JSP file

Session Tracking in JSP

Problem Statement:

You have been asked to explore session tracking in JSP.

Outcome:

Exploring session tracking in JSP enables you to understand how to maintain user state and data across multiple requests during a session. This knowledge is essential for creating personalized and interactive web applications that provide a seamless user experience.

Note: Refer to the demo document for detailed steps: 05_Session_Tracking_in_JSP

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Create a new JSP file
2. Forward the page to the home.jsp page
3. Add cookies in the login.jsp
4. Navigate to login.jsp and add attributes
5. Write an operation to read the data from the URL
6. Write an operation for hidden field data
7. Write an operation for the HTTP session object

JSP and JDBC

Problem Statement:

You have been asked to explore JSP and JDBC.

Outcome:

Exploring JSP and JDBC allows you to understand how to integrate dynamic web pages with database operations. This combination enables the development of data-driven web applications, where user interactions can trigger database queries or updates, enhancing the functionality and interactivity of the web application.

Note: Refer to the demo document for detailed steps: 06_JSP_and_JDBC

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Open the terminal emulator and perform the operation
2. Search for the Maven Repository in the browser
3. Create a new class
4. Create a connection
5. Create another class and pass the attributes
6. Go to the index.jsp page
7. Add a new JSP file
8. Create an object for a prepared statement
9. Create an HTML file in the web app directory
10. Create another JSP file
11. Add an SQL statement
12. Set the if case

Key Takeaways

- JSP is a server-side technology that allows you to create dynamic, platform-independent web applications.
- JSP directive elements provide global information about an entire JSP page.
- JSP action elements are XML-like tags that control the behavior of the servlet engine.
- The JSP Standard Tag Library (JSTL) core tags library provides essential functionality that simplifies the development of JSP-based web applications.

Key Takeaways

- The page directive defines attributes that apply to the JSP page, such as character encoding, scripting language, and error handling.
- JSP scripting elements enable developers to embed Java code directly into HTML pages.
- Expressions output the result of a Java expression directly into the HTML output. The expression is evaluated, converted to a string, and inserted into the HTML.

Thank You