Paul Romer

CSD430

Module 9 Assignment

JSTL and XPath

Jakarta Server Pages (JSP) simplifies web development by combining static HTML with dynamic data from backend processes. Traditionally, JSP used Java scriptlets to embed Java directly into HTML. Although straightforward, scriptlets create several problems. They mix business logic with presentation, reduce readability, and complicate long-term maintenance. They require that page designers to know Java. They conflict with the principle of separation of concerns and the Model-View-Controller (MVC) architecture which can lead to unnecessarily complex, difficult to maintain, and difficult to scale software.

To resolve these issues, the Jakarta Standard Tag Library (JSTL) was introduced. JSTL simplifies JSP development by providing custom tags that replace common scriptlet operations. This library handles typical tasks like managing variables, controlling page flow, formatting data for different locales, accessing databases, and processing XML content. Using tags, JSPs become declarative and clearer, allowing developers to focus purely on presentation.

JSTL naturally aligns with the MVC pattern by clearly separating presentation from business logic. The View (JSP) should display data, not manage it. Typically, data is

prepared by a Controller and stored in JavaBeans—standard Java objects that encapsulate data with getters and setters. JSTL, combined with the Jakarta Expression Language (EL), easily accesses JavaBean properties without Java code. For example, instead of writing scriptlet code like <%= ((UserBean) request.getAttribute("user")).getFirstName() %>, developers can use EL like ${user.firstName}, which directly calls the JavaBean's getter method.

JSTL includes several specialized libraries, each identified by a prefix:

- **Core (c:)**: Fundamental tags like <c:out> for displaying data, <c:set> for managing variables, and <c:forEach> for loops.
- **Formatting (fmt:)**: Tags for formatting numbers (<fmt:formatNumber>) and dates (<fmt:formatDate>).
- **SQL (sql:)**: Basic database access tags (<sql:query> and <sql: update>). This library is best suited for demonstrations or simple use cases. Direct database access in JSP pages is generally discouraged in professional applications.
- **XML (x:)**: XML document processing tags, using XPath for node selection.
- **Functions (fn:)**: Common EL functions for tasks like string manipulation (fn:length, fn:substring).

XML handling is particularly powerful in JSTL, thanks to XPath integration. XPath, standardized by the W3C, is a concise language used to select nodes from XML documents structured as node trees. XPath expressions precisely identify elements, attributes, and text in XML data.

JSTL's XML library (x: prefix) relies heavily on XPath. The <x:parse> tag first processes XML content. Other tags like <x:out>, <x:if>, and <x:forEach> then use XPath through their select attribute to retrieve or control XML data declaratively within JSP. This keeps XML logic clear and separate from Java code.

XML documents can be validated to ensure they conform to predefined structures using Document Type Definitions (DTDs) or XML Schemas. Once validated, JSTL's <x:parse> tag parses XML data into a DOM object that JSP pages can then query using XPath. This helps ensure the data is structured correctly before it is processed by JSP tags.

Historically, JSTL evolved alongside Java EE's transition to Jakarta EE under the Eclipse Foundation. Technical updates included renaming Java packages from javax.servlet.jsp.jstl to jakarta.servlet.jsp.jstl, and updating JSP directive URIs from http://java.sun.com/jsp/jstl/* to jakarta.tags.*. These changes keep JSTL relevant and ensure ongoing alignment with modern Jakarta EE standards.

In day-to-day work, there are several small best practices keep JSTL pages simple, performant, and secure. First, use the <c:choose> family over deeply nested <c:if> blocks. It mirrors switch-case logic and makes the intent obvious to designers reading the page. Second, let the controller populate immutable style beans instead of raw maps. This keeps EL expressions predictable and prevents accidental null-pointer reads. Third, resist the temptation to use the sql: tags too much in production, they are great for prototypes and small projects, but once concurrency and connection-pool tuning become important as you scale, move those queries down into another layer. Finally, remember security.

Everything that eventually renders to the browser should pass through <c:out> or a similar tag so that user-supplied data is automatically HTML-escaped and XSS stays off the table.

In summary, JSTL significantly enhances JSP development by offering a clean, tag-based alternative to scriptlets, simplifying the presentation layer, and enabling conformity to MVC principles. Its integration with XPath provides robust XML handling capabilities, making it an essential tool in creating maintainable, scalable web applications.

Sources

1. https://www.baeldung.com/jstl
2. https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm#:~:text=The%20JavaServer%20Pages%20Standard%20Tag,common%20to%20many%20JSP%20applications.
3. https://en.wikipedia.org/wiki/XPath
4. https://jakarta.ee/specifications/tags/3.0/jakarta-tags-spec-3.0
5. https://www.ibm.com/docs/en/b2b-integrator/6.2.0?topic=extensions-jstl-concepts-definitions
6. https://docs.oracle.com/javaee/1.4/tutorial/doc/JSTL5.html