Noel Miranda

April 28, 2025

Server Side Development

Module 9

JSTL and XPath

When it comes to building dynamic web applications using Java, two tools that stand out for working with the inner components of XML and simplifying the Java Server Pages (JSP) are JSTL and XPath. At first, both may seem confusing for beginners, as was the case for me, but once developers get the hang of how they work together, they can make development way more manageable. This paper explores what JSTL and XPath are, how they interact, and why they are useful in server-side development. It also includes personal reflections on their practicality and relevance to a personal project.

In view of JSTL, it stands for JavaServer Pages Standard Tag Library. It is a collection of custom tags that can be used in JSP to handle common tasks like displaying output, looping through data, or working with XML. It also helps remove Java code from JSP files, making it easier to read and manage. Furthermore, according to Baeldung (2018), JSTL allows developers to focus more on presentation rather than backend logic by offering tags for iteration, conditionals, internationalization, and more. This is especially helpful in team environments where not everyone is comfortable reading Java code directly inside a web page. With JSTL, the page layout remains clean and easier to interpret, appearing more like standard HTML than a mixture of markup and code.

XPath, on the other hand, is a language used to navigate through elements and attributes in an XML document. It serves as a set of instructions for locating specific data within structured files. According to W3Schools (2019), XPath can be used to select nodes, match values, and

even perform basic operations like counting nodes or identifying elements that meet specific criteria. Its flexibility also allows for concise expressions that extract deeply nested information without requiring large amounts of code.

When JSTL and XPath are used together, the result is a powerful method for processing and presenting XML content within JSP files. JSTL includes XML-specific tags that work seamlessly with XPath expressions. According to Minh (2019), JSTL's XML tag library provides tools such as <x:parse>, <x:out>, <x:forEach>, and <x:set>, which allow developers to parse, loop through, and display XML content without writing Java code for each action. To simplify the concept, this paper includes a personal project that mocks a real-world example of a police license plate search system while utilizing both tools. The provided XML file contains mock data about license plates, such as a registered vehicle, owner name, whether the vehicle has been reported stolen, and so on. Instead of writing embedded Java code (scriptlets) to extract and present this data, I was able to use JSTL tags, majorly XML tags and some minor Core tags, with XPath expressions to display it directly in a JSP file. This method makes it feel quite similar to writing HTML, only more dynamic and structured.

It is important to highlight that utilizing this combination of tools also promotes readability and clarity. During the research of this paper, I encountered multiple code examples of other developers working with XML in Java and it involved long blocks of code, nested loops, and complex conditions. However, when I encountered the JSTL code examples, the difference was night and day. The examples demonstrated how developers streamlined the process with JSTL, thus making the codebase easier to read and maintain. It also emphasized how working with both JSTL and XPath aids in development by offering direct paths to the needed data, thus cutting down time spent filtering through XML elements manually. On that note, what really makes XPath useful is its ability to filter results with precision. For instance, referencing the

supplemental code I provided with this paper, I was able to easily retrieve all criminal records involving the offense of ‘speeding’ from the many differing criminal records in the xml file with just the XPath expression //record[offense='Speeding']. This instruction tells the system to extract only those elements that match a specific element value instead of iterating through the whole data set. Therefore, as can be seen in my example and as Santiago (2001) noted in his article, XPath is helpful in building applications that require advanced filtering or searches across large datasets.

I would also like to emphasize how combining JSTL and XPath fits perfectly within a Model-View-Controller (MVC) architecture, a design pattern in Java web development which I recently learned in my server-side development course (Manelli and Zambon, 2020). Basically, this approach keeps the view layer (what users see) separate from the business logic and data models. In practice, this leads to cleaner code organization and easier troubleshooting. When each part of the application is responsible for only one thing, updates and changes become far more manageable.

On another note, JSTL and XPath seem to also promote better long-term habits in web development. When JSP files are written with a mix of Java and markup, I have noticed that they can quickly become difficult to read and maintain. However, by using JSTL tags and XPath expressions, developers write clearer and more structured code. As a result, this can help teams understand each other’s work and improve collaboration.

From a personal perspective, learning how to use JSTL and XPath was a refreshing change. While XML is not as common as JSON in newer applications, it is still very relevant in enterprise systems (Manelli & Giulio Zambon, 2020). Therefore, by understanding these tools, developers are better equipped to maintain and enhance existing systems that rely on XML standards. By building the example for this paper, which involved utilizing JSTL and XPath to

retrieve and display specific data from an XML file, it became clear how practical these tools really are. On top of all that, the process of writing expressions and instantly seeing results made the learning process more interactive.

Overall, JSTL and XPath simplify the process of working with XML in JSP pages, helping developers write cleaner, more efficient, and easier-to-maintain code. They also eliminate the need for embedding Java logic in presentation files, which aligns with best practices for scalable and maintainable dynamic web applications. It is important to note, although XML might not be the newest technology in web development, understanding how to use JSTL and XPath gives developers a useful skill set that can be applied in many real-world situations. I also believe these tools are still worth learning and using, even if one is just trying to understand the interconnected components of Java dynamic web applications.

References

baeldung. (2018, April 12). *A Guide to the JSTL Library | Baeldung*. Www.baeldung.com. https://www.baeldung.com/jstl

Manelli, L., & Giulio Zambon. (2020). *Beginning Jakarta EE web development : using JSP, JSF, MySQL, and Apache Tomcat for building Java web applications*. Apress.

Minh, N. H. (2019). *JSTL XML Tag x:set Example*. Codejava.net. https://www.codejava.net/java-ee/jstl/jstl-xml-tag-set

Santiago, S. (2001, January 26). Combine the power of XPath and JSP tag libraries. InfoWorld. https://www.infoworld.com/article/2159809/combine-the-power-of-xpath-and-jsp-tag-libraries.html

w3schools. (2019). *XPath Syntax*. W3schools.com. https://www.w3schools.com/xml/xpath_syntax.asp