Colton Kohler

7/20/2025

CSD 430 M10 Assignment

**Creating Custom Tags in JSP**

Custom tags in JavaServer Pages (JSP) offer a modular, maintainable, and reusable way to handle server-side logic within web applications. They function as user-defined elements that extend the standard capabilities of JSP, often replacing scriptlets to promote cleaner code. Custom tags rely on associated tag handler classes and are declared using Tag Library Descriptors (TLDs). When used properly, they significantly improve the clarity and flexibility of JSP-based applications.

Custom tags are an alternative to embedding Java logic directly into JSP files. They remove business logic from view templates by pushing it into reusable Java classes. This design promotes better readability and code separation. As noted by GeeksforGeeks, “Custom tags separate the business logic from Java Server Pages… JSP still controls the flow but processing is delegated to a separate java class” (2025). By moving logic into a dedicated handler class, the JSP itself remains cleaner and easier to work with, especially for front-end developers who may not be comfortable writing Java code directly in page templates.

To implement a custom tag, developers need a tag handler class, a TLD file, and a reference to the tag library within the JSP using the taglib directive. The tag handler class is a Java class that either implements interfaces such as Tag, IterationTag, or SimpleTag, or extends support classes like TagSupport, BodyTagSupport, or SimpleTagSupport (Oracle, n.d.). The use of these interfaces determines the behavior of the tag, such as whether it includes a body, iterates over content, or performs a simple action once.

The SimpleTag interface is widely used today due to its streamlined design. Instead of implementing multiple lifecycle methods, a class implementing SimpleTag only needs to override doTag(). As Pandey (2025) explains, “The ‘doTag’ method runs the logic to print ‘Hello, Sachin Pandey!’”. This simplicity makes it more approachable for newer developers while still allowing powerful processing.

TLD files define the metadata for each tag. Each tag is described using sub-elements such as <name>, <tag-class>, <body-content>, and <attribute>. These elements identify the handler class, indicate how the tag handles content, and describe each attribute’s name, type, and whether it is required. The TLD is stored in the WEB-INF folder, ensuring it is not directly accessible from the web. As stated by the DataFlair team, “The tag class of the tld has the complete path of the tag handler so that the Java file can be mapped to the Tag Library Descriptor” (2025).

Once a custom tag is declared and mapped, it can be invoked using the prefix and tag name within a JSP file. Empty tags can be used like <example:hello name= “Sachin”/>, and body tags can be written as <example:repeat>content</example:repeat>. These tags can accept attributes and produce dynamic output directly on the page or as part of another component’s processed body content. According to Oracle (n.d.), custom tags can be used to “produce output,” “iterate over body content,” or even “redirect users to a login page.”

Among the most compelling advantages of custom tags is their reusability. Once a tag is defined and packaged into a library, it can be used across multiple projects without rewriting the logic. This is a huge benefit over scriptlets, which are bound to a specific page and are harder to read and maintain. “Custom tags can be reused across various applications, but it is not possible to reuse the Java code embedded within a JSP using scriptlets” (GeeksforGeeks, 2025).

There are some drawbacks to consider. First, developing and configuring a custom tag library requires a deeper understanding of Java, JSP lifecycles, and web application structure. The added overhead of writing a Java class and defining TLD files may be unnecessary for simple logic. Also, improper implementation—such as unoptimized loops or poor attribute validation—can affect performance and cause bugs.

In my opinion, custom tags should be the standard for any non-trivial server-side logic in JSP. The ability to reuse code, improve page readability, and keep business logic encapsulated in Java classes aligns with good software engineering practices. During development, I’ve found that JSP files using custom tags are easier to update and understand. They also create a clear boundary between presentation and logic. I avoid scriptlets whenever possible, not just for maintainability, but also to keep JSP syntax consistent and clean for future developers on the team.

**References**

DataFlair Team. (n.d.). JSP Custom Tags with Examples. DataFlair. <https://data-flair.training/blogs/jsp-custom-tags/>

GeeksforGeeks. (2025, April 28). Custom Tags in JSP. <https://www.geeksforgeeks.org/java/custom-tags-in-jsp/>

Oracle. (n.d.). Understanding and Creating Custom JSP Tags. <https://docs.oracle.com/cd/E11035_01/wls100/taglib/quickstart.html>

Oracle. (n.d.). Using Custom Tags. <https://docs.oracle.com/javaee/1.4/tutorial/doc/JSPIntro9.html>

Pandey, S. (2025, January 6). Advanced Java — Custom Tags in JSP. Medium. <https://medium.com/@SachinPandeyOnline/advanced-java-custom-tags-in-jsp-da3f324e8ee0>